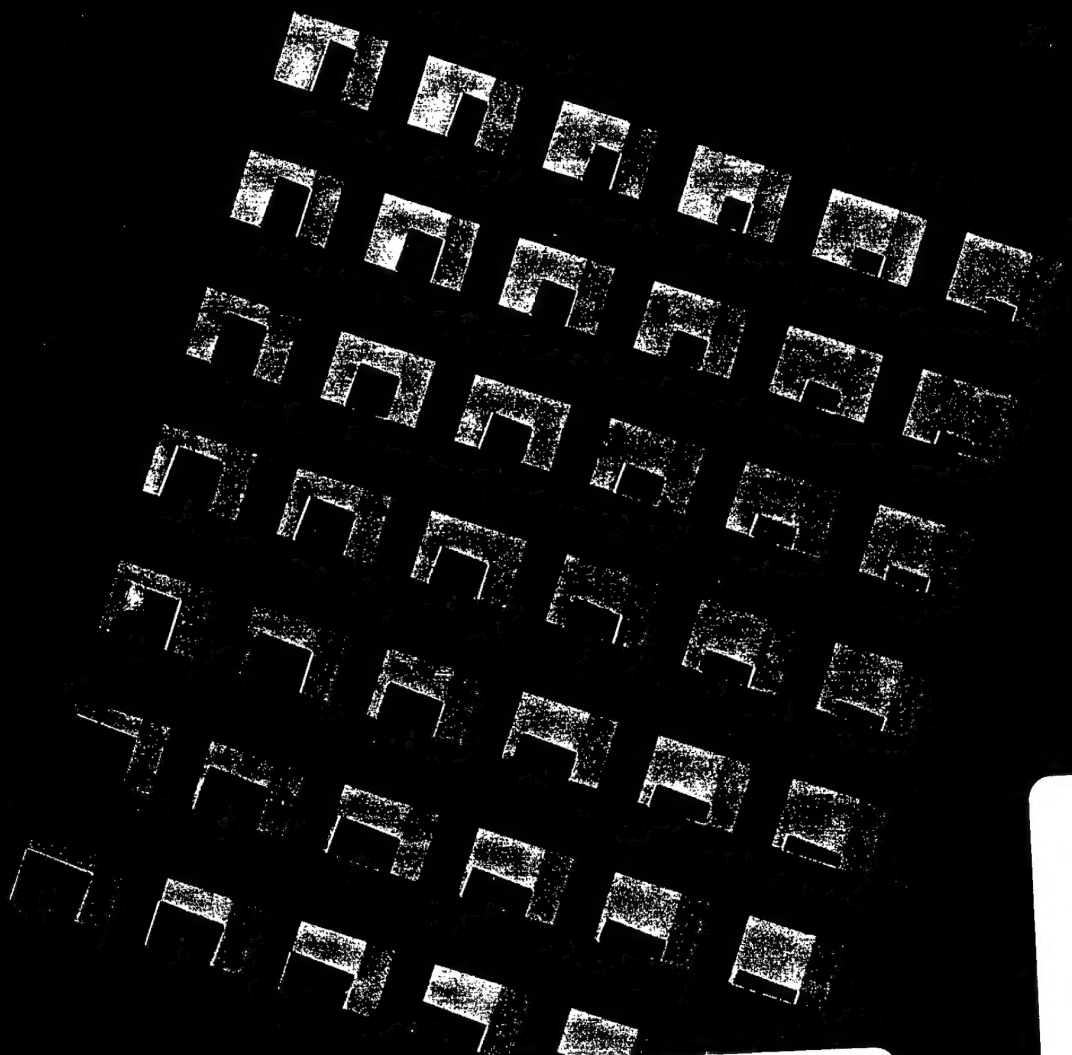
TNO-rapport
FEL-95-A285

IR Reikwijdtevoorspeller - versie 1.0

TNO Fysisch en Elektronisch
Laboratorium



TNO-rapport
FEL-95-A285

IR Reikwijdtevoorspeller - versie 1.0

TNO Fysisch en Elektronisch Laboratorium

Oude Waalsdorperweg 63
Postbus 96864
2509 JG 's-Gravenhage

Telefoon 070 374 00 00
Fax 070 328 09 61

Datum
augustus 1996

Auteur(s)
Ing. M.P. Sannes
Ing. J.C. Vuijst
Dr. A.M.J. van Eijk

Rubricering

Vastgesteld door	:	Drs. W. Pelt
Vastgesteld d.d.	:	30 juli 1996

Titel	:	Ongerubriceerd
Managementuittreksel	:	Ongerubriceerd
Samenvatting	:	Ongerubriceerd
Rapporttekst	:	Ongerubriceerd
Bijlage A	:	Ongerubriceerd

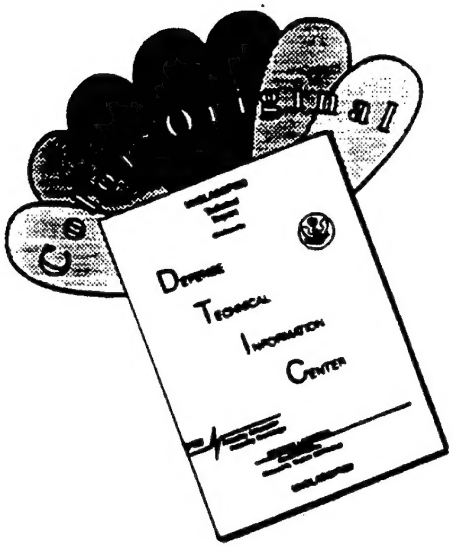
Exemplaarnr.	:	9	
Oplage	:	36	
Aantal pagina's	:	83	(incl. bijlage, excl. RDP & distributielijst)
Aantal bijlagen	:	1	

TNO Fysisch en Elektronisch Laboratorium is onderdeel van de hoofdgroep TNO Defensieonderzoek waartoe verder behoren:

TNO Prins Maurits Laboratorium
TNO Technische Menskunde

Nederlandse Organisatie voor toegepast-natuurwetenschappelijk onderzoek TNO

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF COLOR PAGES WHICH DO NOT REPRODUCE LEGIBLY ON BLACK AND WHITE MICROFICHE.

Managementuittreksel

Titel	:	IR Reikwijdtevoorspeller - versie 1.0
Auteur(s)	:	Ing. M.P. Sannes, Ing. J.C. Vuijst, Dr. A.M.J. van Eijk
Datum	:	augustus 1996
Opdrachtnr.	:	A92KM613
IWP-nr.	:	766.4
Rapportnr.	:	FEL-95-A285

Voor een verantwoorde risico-analyse tijdens maritieme operaties is het noodzakelijk de detectiekans van en de beschikbare reactietijd op dreigingen te kennen. Een belangrijke factor hierbij is het bereik van radar en infrarood waarnemingssystemen in gegeven meteorologische omstandigheden. Dit bereik kan worden berekend met een zgn. reikwijdtevoorspeller.

Een eerste versie van een reikwijdtevoorspeller voor (thermisch) infrarode waarnemingssystemen is ontwikkeld. In deze voorspeller is de binnen het FEL aanwezige kennis op het gebied van propagatie van electro-optische straling door de atmosfeer samengebracht. In dit project is de nadruk gelegd op de rol van de atmosfeer: sensor en doelen zijn slechts schetsmatig gemodelleerd. Deze voorspeller is zonder aanvullend werk dan ook nog niet operationeel bruikbaar.

Op basis van standaard meteorologische parameters (zoals windsnelheid en temperatuur) berekent de reikwijdtevoorspeller de propagatie-eigenschappen van de atmosfeer, waarbij effecten als extinctie door aerosolen en moleculen, alsmede breking (super- en subrefractie) expliciet worden meegenomen.

De reikwijdtevoorspeller is geschikt voor scenario's met een doel op grote afstand (orde 30 km) en lage hoogte (minder dan 50 m). Aparte modules zijn beschikbaar voor de Noordatlantische oceaan en de Nederlandse kust. De gebruiker kan zowel een overzicht krijgen van de detectiekans van doelen in een gegeven scenario, als een indruk krijgen van de invloed van meteorologische parameters op de detectiekans.

Veel aandacht is besteed aan de gebruikersvriendelijkheid van de reikwijdtevoorspeller. Het product draait in een Windows-omgeving, en alle modules zijn volledig geïntegreerd: de gebruiker dient alleen het meteorologisch scenario te specificeren. Een uitgebreide gebruikershandleiding is in dit rapport opgenomen.

Daarnaast worden de diverse modules van de reikwijdtevoorspeller kort beschreven, alsmede een technische beschrijving van het programma gegeven. Tenslotte worden aanbevelingen gedaan voor een volgende versie. De aanbevelingen liggen zowel op het vlak van additionele modules welke de voorspelling verbeteren, als op het gebied van gebruikersgemak.

Inhoud

1. Inleiding

In moderne oorlogvoering is het van het grootste belang om dreigingen in een zo vroeg mogelijk stadium te detecteren. De hoge snelheden die (geleide) projectielen bereiken geven de verdediger slechts luttele seconden de tijd om te reageren. Daarom worden nieuwe generaties waarnemingssytemen met een groot bereik ontwikkeld. Het precieze bereik van deze systemen hangt af van de propagatie-eigenschappen van de atmosfeer, die op hun beurt weer bepaald worden door de meteorologische omstandigheden. Uiteraard spelen ook de karakteristieken van het systeem (b.v. gevoeligheid) en het doel (b.v. grootte) een rol.

Voor een verantwoorde risico-inschatting dient een commandant te weten wat het bereik van zijn waarnemingssystemen is onder de heersende condities. Daarnaast zal hij een inschatting willen maken hoe dit bereik verandert als de weersomstandigheden veranderen (b.v. opstekende wind). Hiervoor is een reikwijdtevoorspeller nodig: een model dat aan de hand van meteorologische, systeem- en doelparameters het bereik van een waarnemingssysteem voorspelt. In een reikwijdtevoorspeller zijn dus drie afzonderlijke modules te onderscheiden: een atmosferische, sensor en doelmodule.

De atmosferische module van een reikwijdtevoorspeller voor infrarode (thermische) waarnemingssystemen dient rekening te houden met extinctie, turbulentie en refractie in de onderste lagen van de atmosfeer. Extinctie ten gevolge van absorptie en verstrooiing door aerosol en moleculen vermindert het contrast van het doel met zijn achtergrond. Turbulentie veroorzaakt kleine veranderingen in de brekingsindex waardoor scintillatie en "beam wander" ontstaan. Refractie treedt op door een variatie in de gemiddelde brekingsindex als functie van de hoogte en leidt tot "ray-bending", waardoor de afstand tot de optische horizon verandert en hetzelfde doel onder twee verschillende hoeken kan worden waargenomen. Een gedetailleerde beschrijving van deze verschijnselen wordt gegeven door De Leeuw et al. [1995a,b].

In het kader van opdracht A92KM613 is een eerste versie ontwikkeld van een reikwijdtevoorspeller voor infrarode (thermische) waarnemingssystemen. Hierbij is de nadruk gelegd op de atmosferische module: sensor en doel zijn slechts schetsmatig gemodelleerd. De reikwijdtevoorspeller is vooral gericht op scenario's met een dreiging vlak boven het zee-oppervlak.

In deze reikwijdtevoorspeller zijn bestaande fysische en empirische modellen die de propagatie-eigenschappen van de atmosfeer beschrijven samengebracht. Bij de ontwikkeling van deze modellen is gebruik gemaakt van de grote hoeveelheid meetgegevens die in de afgelopen 15 jaar verzameld is. Voor de ontwikkeling van deze eerste versie van de infrarood reikwijdtevoorspeller zijn met name gegevens verkregen tijdens de Cumulus meetcampagne [De Leeuw et al., 1986], het

HEXMAX experiment [Smith et al., 1990] en het MAPTIP programma [De Leeuw et al., 1994] van groot belang geweest.

Bij de ontwikkeling hebben twee randvoorwaarden een grote rol gespeeld. In de eerste plaats dient de reikwijdtevoorspeller gebruik te maken van standaard meteorologische parameters, zoals deze aan boord van een operationeel schip beschikbaar zijn. Hierbij moet gedacht worden aan parameters als windsnelheid, zee- en luchttemperatuur, druk en vochtigheid. Ten tweede dient de reikwijdtevoorspeller gebruikersvriendelijk te zijn.

In hoofdstuk 2 wordt een korte beschrijving gegeven van de diverse modellen die in de reikwijdtevoorspeller zijn opgenomen. Voor gedetailleerde informatie wordt verwezen naar de referenties. Hoofdstuk 3 bevat een gebruikershandleiding voor de reikwijdtevoorspeller. Een technische beschrijving van de code en de programmeertaal wordt gegeven in hoofdstuk 4. Tenslotte worden aanbevelingen gedaan voor een volgende versie van de reikwijdtevoorspeller.

2. Modelbeschrijving

In dit hoofdstuk wordt een beknopte beschrijving gegeven van de diverse modellen die in de reikwijdtevoorspeller verwerkt zijn. Het betreft hier het micrometeorologisch model, het model voor moleculaire extinctie, de modellen voor aerosol extinctie op dekhoogte, het model voor het verticale profiel van de aerosol extinctie, en het model voor "ray-tracing". Voor een uitgebreide beschrijving van de modellen wordt verwezen naar de referenties die bij elk model genoemd worden.

2.1 Micrometeorologisch model

Het micrometeorologisch model wordt aangeroepen om een aantal micrometeorologische grootheden, zoals ruwheidslengte en schalingsparameters, uit te rekenen [zie o.a. Stull, 1988]. Deze parameters worden vervolgens gebruikt om de waarden van windsnelheid, relatieve vochtigheid en luchttemperatuur, zoals gemeten door de sensoren op een bepaalde hoogte, te converteren naar de waarden voor de standaardhoogte van 10 meter boven gemiddeld zeeniveau. In versie 1.0 is één micrometeorologisch model geïmplementeerd, het LKB-bulk model [Liu et al., 1979].

Het LKB model maakt gebruik van de zgn. "drag-coefficient", die de relatie geeft tussen de windsnelheid op 10 meter hoogte en de frictie snelheid aan het oppervlak (zie Geernaert [1990] voor een uitgebreide discussie). In versie 1.0 zijn twee relaties voor de "drag-coefficient" geïmplementeerd: een relatie gegeven door Smith [1980] voor "open ocean" condities, en een relatie gegeven door Davidson [1995] voor de Noordzee (gebaseerd op het HEXMAX experiment, zie Smith et al. [1990]). Het LKB-model kiest uit deze twee relaties afhankelijk van het gekozen aerosol model (zie onder): Noordzee voor het MPN-model, "open ocean" voor het NAM- en TNO-open ocean model.

Het LKB-model berekent de atmosferische schalingsparameters (u*, Q* en T*), de ruwheidslengte z_0, en de Monin-Obhukov lengte L, welke een maat is voor de atmosferische stabiliteit. Met deze parameters kunnen windsnelheid, luchttemperatuur en vochtigheid op een willekeurige hoogte worden berekend volgens een logarithmisch profiel, gecorrigeerd voor de stabiliteit.

Het LKB-model werkt niet in een zeer stabiele atmosfeer, d.w.z., een warme luchtmassa boven een koude zee in combinatie met een lage windsnelheid. De precieze grenzen zijn afhankelijk van de windsnelheid: voor een windsnelheid lager dan 5 m/s en een temperatuursverschil tussen lucht en zee groter dan 5 °C werkt het model niet. Een uitgebreide discussie wordt gegeven door Smith [1988].

2.2 Moleculair model

De moleculaire extinctie wordt uitgerekend met een EO/IR atmosferisch transmissiemodel dat ontwikkeld is door LeClerc [1989] van DREV (Canada). Dit model is gebaseerd op een parameterisatie van de Lowtran 6 code [Kneizys et al. 1983] en geldt onder de aanname van uniforme spectrale radiantie. In dat geval kan de moleculaire extinctie wordt berekend als een som van een golflengte onafhankelijk continuum en een golflengte afhankelijke absorptie.

Het model heeft drie meteorologische parameters nodig: de relatieve vochtigheid, atmosferische druk en luchttemperatuur. De moleculair extinctie kan berekend worden voor een viertal laserlijnen (10.6, 3.8, 1.06 en 0.6 μm) en een drietal banden (0.6-1.0, 3-5 en 8-12 μm). Uit testen is gebleken dat een correcte berekening vooral afhangt van een juiste waarde voor de luchttemperatuur (zie [Schwering en Van Eijk, 1994]). Elke 0.2 °C afwijking leidt tot 1% afwijking in de extinctie.

LeClerc [1989] valideert het model t.o.v. Lowtran 6 aan de hand van drie atmosferische verticale profielen waarin vochtigheid, druk, en luchttemperatuur sterk variëren. Afwijkingen t.o.v. Lowtran 6 zijn typisch in de orde van enkele procenten over padlengtes van 30 km.

2.3 Aerosol modellen

De aerosol extinctie wordt uitgerekend in twee stappen. In de eerste stap wordt de extinctie op dekhoogte (10 meter boven zeeniveau) uitgerekend (zie onder), waarna het verticale profiel van de aerosol extinctie wordt uitgerekend (zie volgende paragraaf). Versie 1.0 van de reikwijdtevoorspeller beschikt over drie aerosol modellen om de extinctie op dekhoogte uit te rekenen: een algemeen model (NAM), een model voor de noordelijke Atlantische Oceaan, en een model specifiek voor de regio rond het MPN-platform (Noordzee).

2.3.1 Navy Aerosol Model (NAM)

Het Navy Aerosol Model (NAM) is ontwikkeld door Gathman [1983]. De reikwijdtevoorspeller gebruikt een versie [Hughes, 1987; Gathman, 1989] die verbeterd is aan de hand van theoretisch [Gerber, 1985] en experimenteel onderzoek [De Leeuw, 1986]. NAM is een algemeen aerosol model dat gebaseerd is op een grote set meetgegevens afkomstig uit diverse geografische gebieden. Het is vooral geschikt voor "open ocean" scenario's, maar mag ook gebruikt worden voor kustgebieden indien de invloed van het land gering is. Het NAM model is minder geschikt voor binnenzeeën zoals de Noordzee en voor sterk verontreinigde gebieden [Tanguy et al., 1991; Van Eijk en De Leeuw, 1992a].

Behalve standaard meteorologische parameters zoals windsnelheid en relatieve vochtigheid, heeft het NAM-model twee parameters nodig die mogelijk niet standaard aanwezig zijn. Dit zijn de gemiddelde windsnelheid in de afgelopen 24 uur en het zicht bij 0.55 μm. NAM gebruikt deze laatste parameter om de zgn. "air mass parameter" uit te rekenen, die de herkomst (continentaal of maritiem) van de luchtmassa parametriseert.

Het NAM model heeft echter als voordeel dat het in vrijwel alle meteorologische omstandigheden kan worden toegepast. De enige restrictie die in de praktijk zal voorkomen, is dat de relatieve vochtigheid niet boven de 99% mag komen. Een tweede voordeel van NAM is dat het niet alleen de aerosol extinctie voor een aantal laserlijnen kan uitrekenen, maar ook bandextincties (zie paragraaf Betrouwbaarheid).

2.3.2 TNO Open Ocean model

Het TNO Open Ocean model is uitgebreid beschreven door De Leeuw et.al. [1993]. Het model is gebaseerd op metingen die in mei-juni 1983 nabij het Ocean Weather Station (OWS) Lima (57 °N, 20 °W) in de Noord-Atlantische oceaan zijn verricht (zie De Leeuw et.al. [1986]). Het model is dan ook niet geschikt voor ander geografische gebieden.

De input voor het model bestaat in ieder geval uit de windsnelheid. Een betere voorspelling wordt verkregen als ook relatieve vochtigheid, luchttemperatuur en zeewatertemperatuur bekend zijn. Het model kan niet gebruikt worden in meteorologische scenario's met een hoge windsnelheid (meer dan 15 m/s). De betrouwbaarheid van het model neemt af naarmate de luchttemperatuur toeneemt (hoger dan 15 °C), en naarmate de atmosfeer onstabieler wordt (ASTD < -5 °C) (zie ook Schwering en Van Eijk [1994]).

Het TNO Open Ocean model kan de aerosol extinctie berekenen voor 5 laserlijnen, te weten 10.6, 4.0, 1.064, 0.6943 and 0.55 μm. De berekende extinctie wijkt gemiddeld een factor 1.5 af van de werkelijke extinctie. (68% confidence limit) [De Leeuw et.al., 1993].

2.3.3 MPN Aerosol model

Het MPN aerosol model [Van Eijk en De Leeuw, 1992a,b] is gebaseerd op meetgegevens die tijdens het HEXMAX-experiment in october-november 1986 op het MPN-platform in de Noordzee zijn verzameld [Smith et al., 1990]. Het model is gevalideerd met data verkregen tijdens het MAPTIP experiment [Van Eijk et al., 1994]. Het is één van de weinige modellen die de aerosol extinctie in een kustgebied beschrijven, maar het is helaas uitsluitend bruikbaar binnen een straal van 40 km rondom het MPN platform.

De input parameters van het model zijn windsnelheid en windrichting. De betrouwbaarheid van de voorspelling neemt toe als ook relatieve vochtigheid en

ASTD gespecificeerd zijn. Het model is bruikbaar voor windsnelheden tot 25 m/s. De betrouwbaarheid van het model neemt af wanneer de windrichting tussen 0° en 90° N ligt en/of wanneer de luchttemperatuur hoger is dan 20 °C [Schwering en Van Eijk, 1994].

Het MPN aerosol model kan de extinctie berekenen voor 5 laserlijnen, te weten 10.6, 4.0, 1.064, 0.6943 and 0.55 μm. De berekende extinctie wijkt gemiddeld een factor 2.0 af van de werkelijke extinctie. (68% confidence limit) [Van Eijk en De Leeuw, 1992c].

2.3.4 Betrouwbaarheid aerosol band extinctie

De Open Ocean en MPN aerosol modellen kunnen geen bandextinctie uitrekenen. Daarom wordt de aerosol extinctie in de 3-5 μm band benaderd met de 4 μm lijnextinctie. Deze benadering is toegestaan indien de aerosol extinctie niet sterk afhankelijk is van de golflengte. Deze hypothese is getest met het NAM model, dat zowel band- als lijnextinctie kan berekenen.

De 4 μm lijnextinctie bleek ongeveer 5% kleiner te zijn dan de 3-5 μm bandextinctie wanneer het zicht groter was dan 20 km. Als het zicht afneemt, neemt de discrepantie tussen lijn- en bandextinctie snel toe. Bij een zicht van 5 km is de afwijking 30%, bij een zicht van 100 m is dit 60%.

Op dezelfde wijze werd ook het benaderen van de 8-12 μm bandextinctie met de 10.6 μm lijnextinctie getest. Hier bleek de lijnextinctie 12% kleiner dan de bandextinctie. Bij lage windsnelheden en/of hoge relatieve vochtigheid loopt de onderschatting op tot 20%.

De precieze afwijking tussen lijn- en bandextinctie is afhankelijk van de sensor karakteristieken. In de hierboven beschreven testen is gelijk gewicht toegekend aan elke golflengte in de band (statistisch gemiddelde). In werkelijkheid zal de sensorgevoeligheid moeten worden meegenomen, hetgeen de percentages zoals hierboven gegeven kan veranderen (met name in de 3-5 μm band).

2.4 Verticale profiel van aerosol extinctie

Het verticale profiel van de aerosol extinctie wordt uitgerekend met een model van Goroch et al. [1980] en De Leeuw [1989]. Dit model is echter ontwikkeld om de verticale verdeling van aerosolen van een bepaalde diameter te berekenen. In principe dient dus de verticale verdeling voor een groot aantal diameters te worden doorgerekend, waarna op elke hoogte m.b.v. Mie-theorie de extinctie kan worden berekend.

In deze eerste versie van de reikwijdtevoorspeller wordt echter aangenomen dat het verticale profiel voor aerosolen met een (droge) straal van 1 μm representatief is

voor het verticale profiel van de aerosol extinctie. Deze straal is gekozen omdat deze redelijk overeenkomt met de gemiddelde (natte) straal van het aerosol. Deze benadering is echter nog niet gevalideerd. Davidson et al. [1994] hebben laten zien dat het model de verticale verdeling van het aerosol goed kan berekenen, behalve in kustgebieden met aflandige wind.

Het verticale profiel wordt berekend aan de hand van micrometeorologische parameters (Monin-Obhukov lengte, schalingsparameters en ruwheidslengte), die beschikbaar zijn uit het LKB model (zie boven). Effecten van de relatieve vochtigheid (verticaal profiel, groeien en krimpen van het aerosol) worden expliciet gemodelleerd. Entrainment aan de bovenkant van de grenslaag wordt beschreven met modellen van Tennekes en Driedonks [1980] voor een stabiele atmosfeer, van Tennekes [1973] voor een instabiele atmosfeer met vrije convectie, en van Deardorff [1976] voor een instabiele atmosfeer met geforceerde entrainment.

2.5 Refractiemodel

Als gevolg van variaties in de brekingsindex van de atmosfeer worden lichtstralen gebroken. De resulterende stralengang kan worden berekend met een "ray-tracing" model. In deze versie van de reikwijdtevoorspeller wordt een model van Kunz [1995] gebruikt.

Het "ray-tracing" model berekent het verticale profiel van de brekingsindex uit de profielen van temperatuur, vochtigheid en druk. De eerste twee zijn beschikbaar uit het LKB-model (zie boven), de laatste volgt uit de hydrostatische vergelijking. De brekingsindex wordt berekend volgens Edlén [1966]. Deze methode is geldig voor een groot golflengtegebied [Fenn et al., 1985] en stemt goed overeen met vergelijkbare modellen [Kunz, 1993].

Met behulp van het profiel van de brekingsindex kan de stralengang door de atmosfeer worden berekend. In het algemeen geldt dat in een stabiele atmosfeer (lucht warmer dan zee) superrefractie optreedt. Hierdoor is de optische horizon minder ver dan de geometrische horizon en zullen laagvliegende doelen eerst laat worden gedetecteerd. In een instabiele atmosfeer treedt subrefractie op ("ducting"), waardoor de optische horizon juist verder ligt dan de geometrische horizon.

3. Gebruikershandleiding

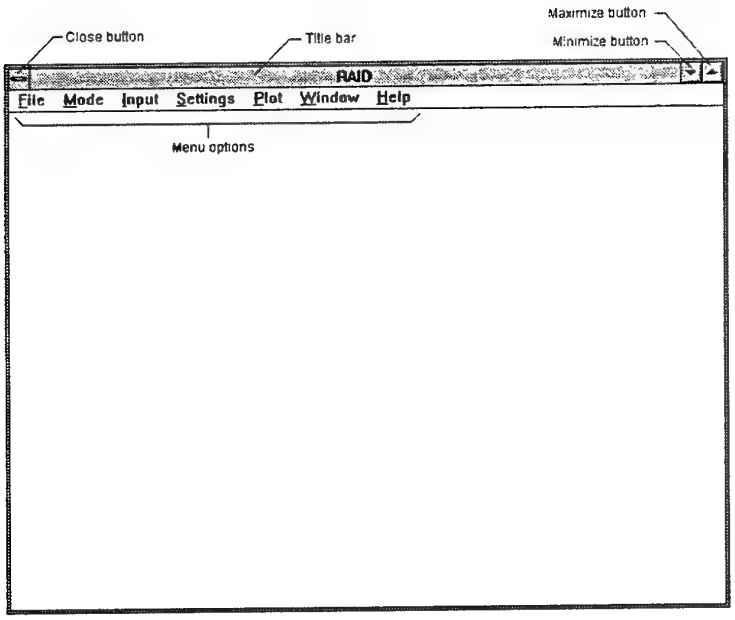
3.1 Systeem eisen

Om RAID op een doeltreffende manier te gebruiken zijn de volgende minimale systeem eisen van toepassing:

- *Computer*: een standaard PC met een 80486 of compatible microprocessor. Microsoft Windows V3.1 of Windows 95. Een harddisk en een 1.44 MB floppy disk drive.
- *Diskruimte*: om RAID te installeren is 1 MB diskruimte nodig. Tijdens het werken met RAID is 30 MB harddisk ruimte nodig.
- *RAM*: om RAID te kunnen gebruiken is 8 Mb RAM nodig.
- *Display*: VGA.
- *Muis*: een Microsoft compatible muis is vereist.
- *Printer*: optie.

3.2 Opties en mogelijkheden van het programma

Zodra RAID wordt gestart, verschijnt een leeg window met een menu bar waarin zich de hoofdmenu opties bevinden. Figuur 3.1 geeft aan uit welke delen dit window is opgebouwd. Selecteer de diverse menu opties om alle RAID commando's te zien. De beschreven buttons van dit figuur zijn in elk plot-window van RAID terug te vinden.

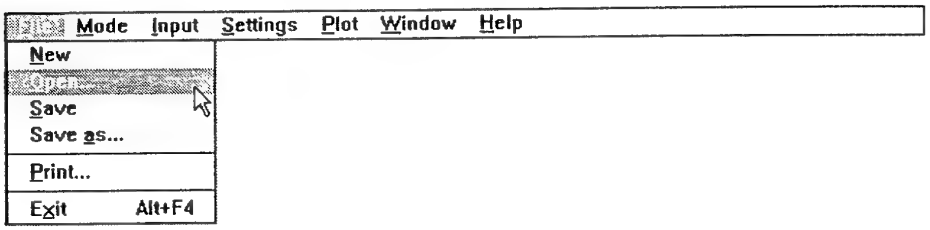


Figuur 3.1: RAID

In het nu volgende deel van de handleiding zal aandacht worden besteed aan diverse mogelijkheden van RAID. Deze zullen worden besproken aan de hand van de aanwezige hoofdmenu opties van dit window.

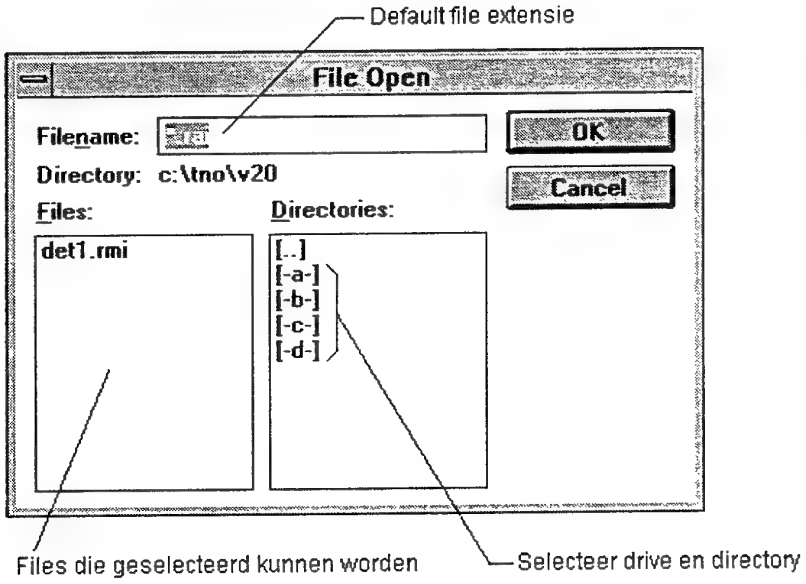
3.2.1 File Menu

Het File menu bestaat uit 6 opties (figuur 3.2):



Figuur 3.2: File Open

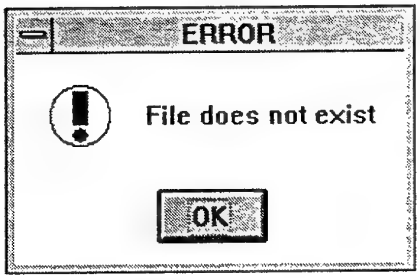
De menu opties *New, Open, Save* en *Save as* hebben betrekking op de files die informatie bevatten over instellingen van aanwezige dialogboxen binnen RAID. Menu optie *New* zet alle invoerveld-instellingen van RAID terug op default waarden. Dit zijn de instellingen waarmee het programma wordt opgestart.



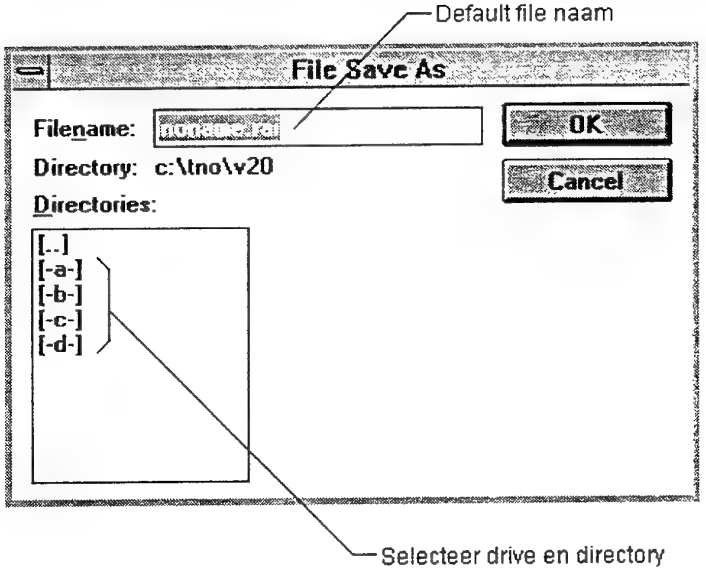
Figuur 3.3: File Open dialogbox

Met menu optie *Open* is het mogelijk om dialogbox-instellingen te lezen van disk. Op deze wijze zijn veel gebruikte combinaties van bepaalde typen sensoren, doelen en/of meteorologische scenario's makkelijk toegankelijk. Door middel van de 'File Open' dialogbox uit figuur 3.3 is het mogelijk een dergelijk bestand op te halen van disk, dat alle instellingen bevat die RAID nodig heeft voor een bereke-

ning. Deze instellingen betreffen alle aanwezige checkboxen, dialogboxen, invoervelden en menu's.
Zodra een bestand niet gelezen of gevonden kan worden, wordt hier een melding van gemaakt zoals is weergegeven in figuur 3.4.



Figuur 3.4: File Error messagebox

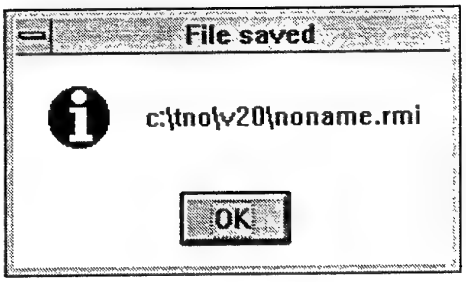


Figuur 3. 5: File Save / File Save as dialogbox

Door middel van de 'File Save / File Save as' dialogbox uit figuur 3.5 is het mogelijk een bestand te schrijven naar disk. Dit bestand bevat alle dialogbox instellingen (zoals instellingen van checkboxen, dialogboxen, invoervelden en instellingen van het menu) van RAID. Alhoewel dit bestand in ASCII is opgeslagen wordt het afgeraden om hierin zelf veranderingen aan te brengen, omdat het format van het bestand aan bepaalde eisen dient te voldoen.

Zodra een file naar disk is geschreven, wordt dit bevestigd met een messagebox (figuur 3.6). File-namen die langer zijn dan 8 karakters worden *automatisch inge-*

kort, zodat altijd een file-naam met een maximale lengte van 8 karakters naar disk wordt geschreven.

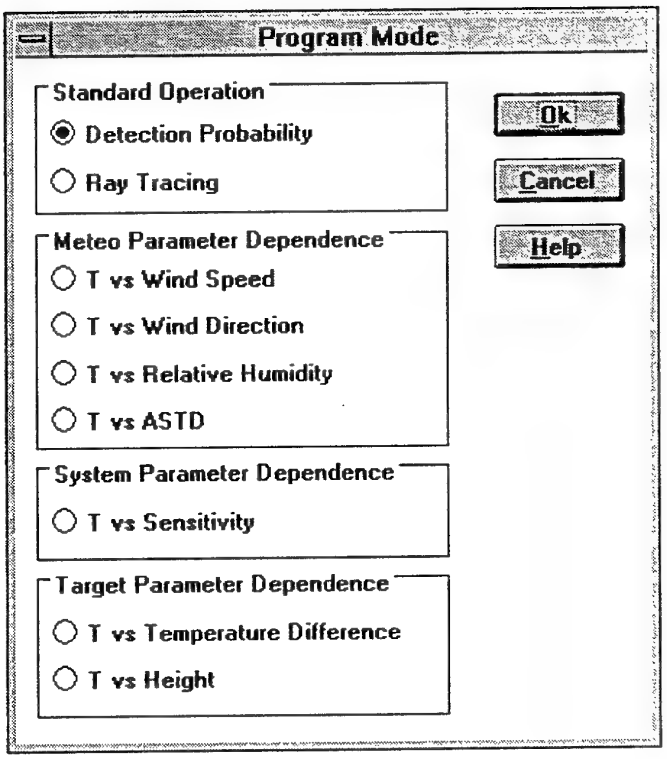


Figuur.3.6: File saved messagebox

Menu optie *Print* is in deze versie van RAID niet geïmplementeerd.
Menu optie *Exit* dient om RAID te beëindigen. Deze optie is ook te verwezenlijken met een *hot key* (Alt+F4).

3.2.2 Mode menu

Met menu optie *Mode* kan de gewenste type berekening worden ingesteld (figuur 3.7). De geselecteerde radio-button geeft aan welke berekening zal worden uitgevoerd. De berekening wordt gestart door hoofdmenu optie *Plot* in de menu bar te selecteren.



Figuur 3.7: Program Mode dialogbox

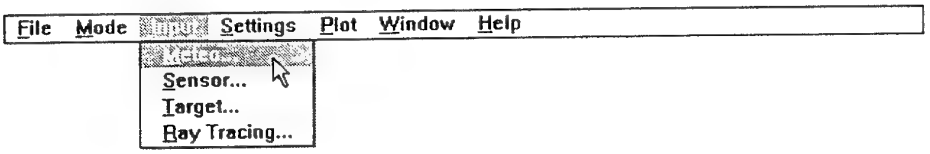
Program Mode is in vier categorieën opgesplitst:

- Standard operation
- Meteo Parameter Dependence
- System Parameter Dependence
- Target Parameter Dependence

In de paragraaf Plot (3.3.5) zullen aan de hand van (plot-)resultaten de diverse modes nader worden besproken. De modes: *T vs Sensitivity, T vs Temperature Difference en T vs Height* zijn in deze versie van RAID niet geïmplementeerd. De *Help* optie binnen de dialogbox is eveneens nog niet geïmplementeerd

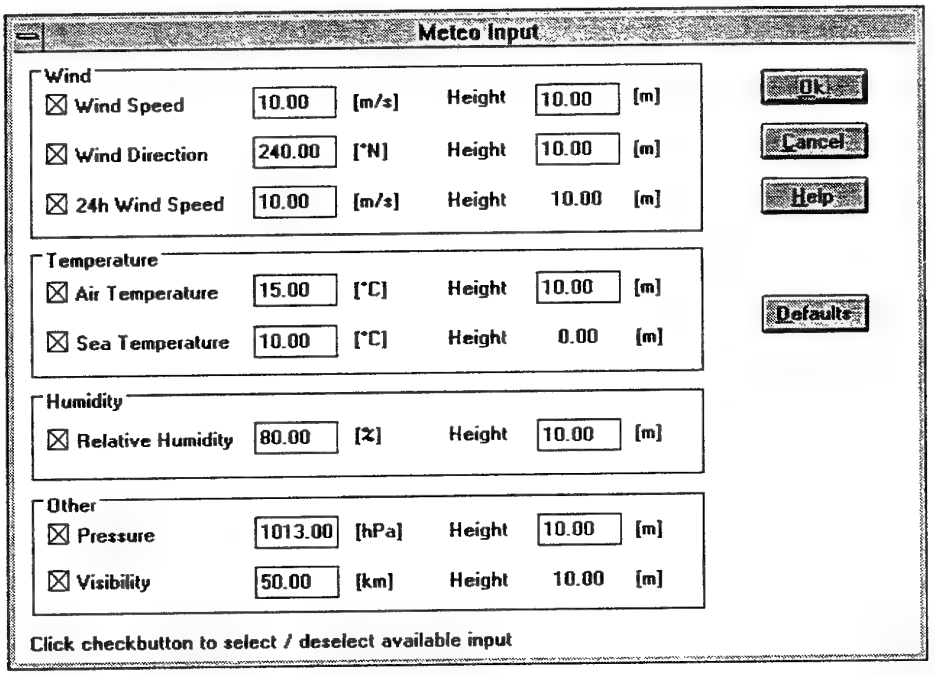
3.2.3 Input menu

Het Input menu bestaat uit 4 opties (figuur 3.8):



Figuur 3.8: Input Meteo

De menu opties *Meteo, Sensor, Target* en *Ray tracing* hebben betrekking op de input parameters die nodig zijn voor een berekening.



Figuur 3.9: Meteo Input dialogbox

In figuur 3.9 is de (default) Meteo Input dialogbox weergegeven. Deze dient om meteorologische parameters in te voeren. De parameters worden gebruikt als input

voor de diverse atmosferische modellen. Er kunnen vier categorieën in de Meteo Input dialogbox worden onderscheiden:

- Wind
- Temperatuur
- Vochtigheid
- Overige

Iedere categorie bevat checkboxen en invoervelden voor het respectievelijk selecteren en invoeren van meteorologische parameters. Voor een beschrijving van deze parameter-invoer wordt als voorbeeld de windsnelheid genomen (figuur 3.10).

Figuur 3.10: Windsnelheid

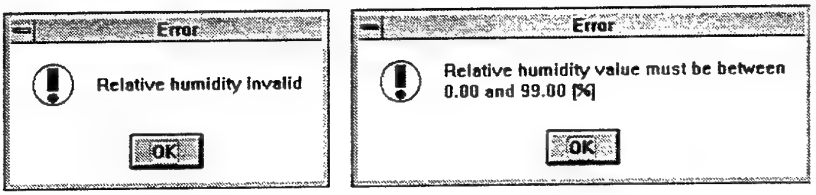
Met de aanwezige checkbox is het mogelijk om aan te geven of een meteorologische parameter bekend is of niet. Bij het uitschakelen van de checkbox komt in de betreffende invoerveld(en) het *getal -99* te staan. Binnen de groep atmosfeerfysica van het FEL codeert dit getal voor een ontbrekende of onbekende parameter. Bij de daadwerkelijke berekening vindt een parameter contrôle plaats, waarbij gekeken wordt of voor de geselecteerde modellen voldoende input parameters beschikbaar zijn. Als dit niet het geval is, wordt hiervan een melding gegeven. Een beschrijving van deze contrôle wordt gegeven in paragraaf 3.3.5.

Door met de muis te klikken op het *Wind Speed* invoerveld is het mogelijk om de windsnelheid te veranderen. De hoogte waarop deze windsnelheid is gemeten, wordt ingevoerd door met de muis te klikken op het *Height* invoerveld. De eenheden waarin de fysische parameters dienen te worden gespecifieerd, zijn rechts naast de invoervelden weergegeven.

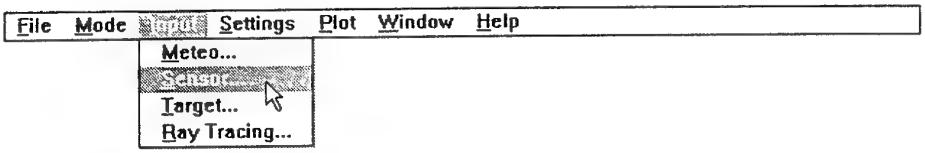
De overige parameters zijn op bovenstaande manier te wijzigen. Het is bij de parameters *24hWindspeed*, *Sea Temperature* en *Visibility* niet mogelijk om de hoogtes te veranderen waarop deze parameters gemeten zijn.

Push-button Ok bevestigt de invoer en push-button Cancel annuleert de veranderingen. Met push-button *Defaults* worden alle checkboxen en controlfields van deze dialogbox terug gezet in hun oorspronkelijke staat. De Help optie binnen deze dialogbox is in deze versie van RAID nog niet geïmplementeerd.

Onjuiste invoer in de Meteo Input dialogbox resulteert in twee soorten message boxen. Als voorbeeld zijn in figuur 3.11 message boxen weergegeven die horen bij het geval dat de invoer voor de relatieve luchtvochtigheid niet als getal kan worden geïnterpreteerd (links) en het geval dat de opgegeven relatieve luchtvochtigheid niet binnen het vereiste domein valt (rechts). Dit domein dient ter voorkoming van onrealistische meteorologische scenario's.

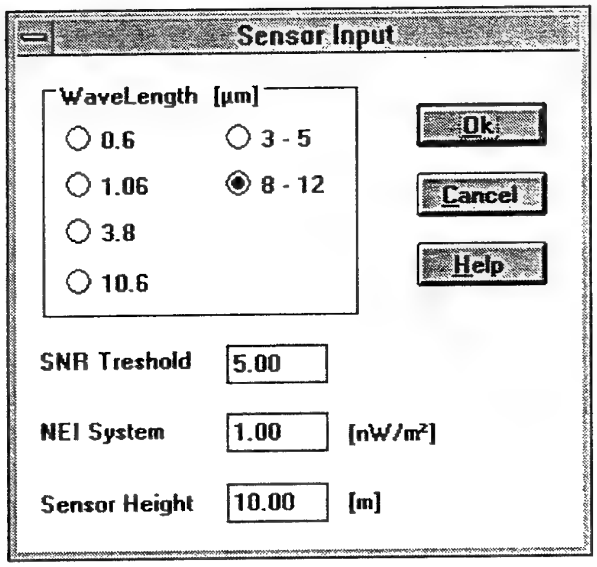


Figuur 3.11: Onjuiste meteo-invoer



Figuur 3.12: Input Sensor

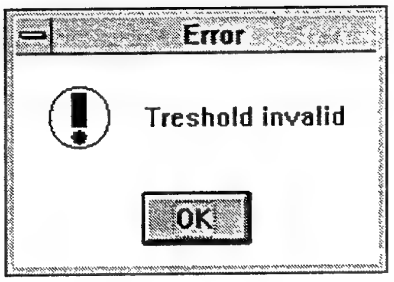
Met menu optie *Sensor* (binnen hoofdmenu optie *Input*, zie figuur 3.12) kunnen gegevens ingevoerd worden die betrekking hebben op de fysische parameters van de sensor. Deze gegevens zijn belangrijk omdat ze de kans op detectie beïnvloeden. Gegevens over de golflengte, de SNR treshold, NEI System (nog niet geïmplementeerd) en ook de hoogte van de sensor worden hier ingebracht. Zodra Ok is ingedrukt hebben de parameters effect. Onderstaande dialogbox, figuur 3.13 toont de in te voeren sensor parameters



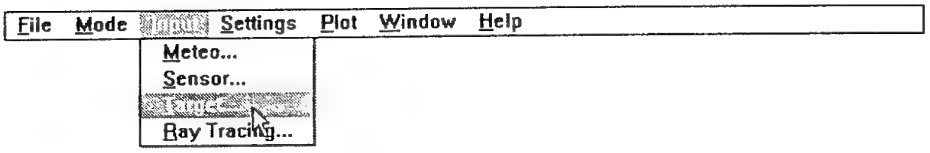
Figuur 3.13: Sensor parameters

Zodra foutieve invoer gegeven wordt, verschijnt na het indrukken van de Ok button een error messagebox. Deze geeft aan waar de fout zich bevindt. Als bijvoorbeeld in het *treshold* input veld een letter ingevoerd wordt, verschijnt een messagebox zoals getoond in figuur 3.14. Als waarden ingevoerd worden die buiten zinvolle limieten van de sensor vallen, verschijnt een messagebox die vermeld binnen welke grenzen de invoer geldig is. De gebruiker kan pas verder als

de invoer correct is. De Cancel button zorgt ervoor dat deze dialogbox verlaten kan worden zonder dat input parameters veranderd zijn.

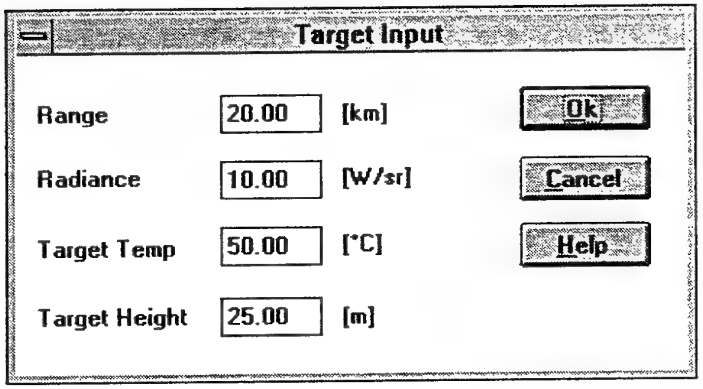


Figuur 3.14: Foutieve invoer



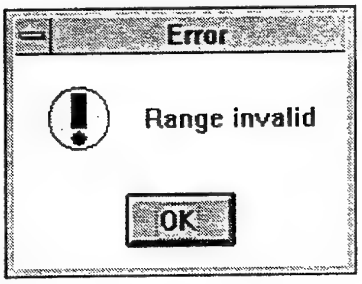
Figuur 3.15: Input Target

Menu optie *Target* binnen de hoofdmenu optie Input bevat de parameters van het doel (figuur 3.15). Parameters zijn de afstand, radiantie, temperatuur en hoogte van het doel. De dialogbox wordt getoond in figuur 3.16.

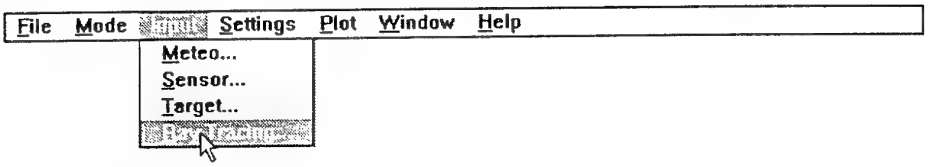


Figuur 3.16: Target parameters

Ook hier wordt foute invoer direkt beantwoord met een error messagebox die vermeld waar de fout zich bevindt. In figuur 3.17 is deze messagebox te zien in een situatie waarbij in het *Range* invoerveld een ongeldig getal is ingevoerd. Bij invoer buiten de grenswaarden verschijnt een messagebox die weergeeft wat de grens-waarden zijn.

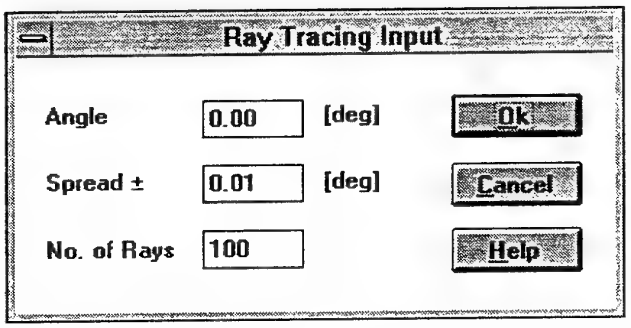


Figuur 3.17: Ongeldig getal

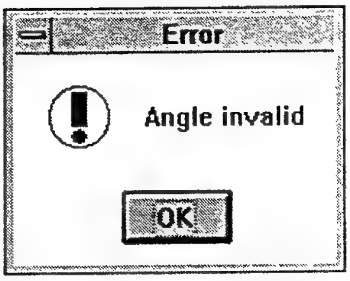


Figuur 3.18: Input Ray Tracing

De laatste menu optie van *Input* is *Ray Tracing* (figuur 3.18). Met deze optie worden parameters gespecificeerd voor een grafische weergave van het verloop van 1 of meerdere lichtstralen naar de sensor toe. Figuur 3.19 toont de input dialogbox en de diverse invoervelden: een hoek waaronder de lichtstraal vertrekt, de spreiding om deze hoek, en het aantal lichtstralen. Het aantal lichtstralen dient zorgvuldig gekozen te worden, omdat de rekentijd sterk toeneemt naarmate meer lichtstralen getoond moeten worden. Wanneer te veel lichtstralen getoond moeten worden, kan het voorkomen dat individuele stralen haast niet meer te zien zijn. Een bijna zwart vlak in de plot kan dan optreden.



Figuur 3.19: Ray Tracing dialogbox

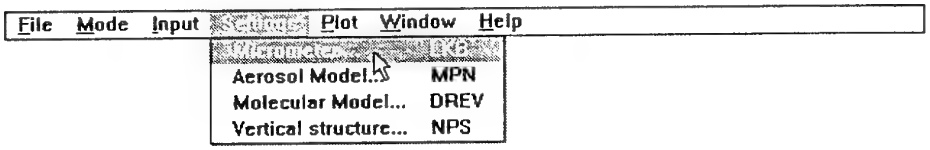


Figuur 3.20: Foutieve invoer

Als foutieve invoer parameters worden ingevoerd, verschijnt een messagebox als figuur 3.20. Gegevens buiten de grenzen worden weergegeven in een messagebox die vermeld binnen welke grenzen invoer moet worden gegeven.

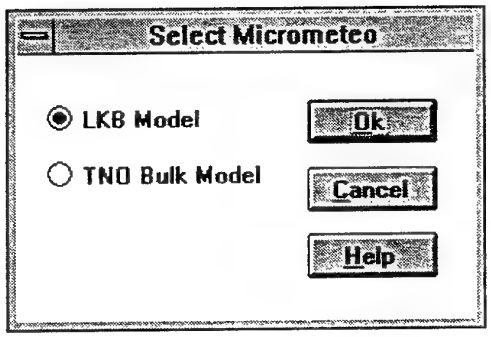
3.2.4 Settings

Het Settings menu bestaat uit 4 opties:



Figuur 3.21: Settings Micrometeo

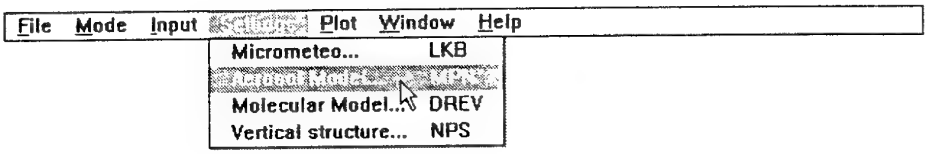
De menu opties *Micrometeo*, *Aerosol Model*, *Molecular Model* en *Vertical structure* hebben betrekking op de selectie van de diverse modellen die gebruikt zijn voor het maken van een berekening. Het *Settings* menu geeft tevens een overzicht van de op dit moment geselecteerde modellen (zie figuur 3.21).
De eerste keuze betreft het micrometeorologisch model dat de schalingsparameters van wind, temperatuur en vochtigheid berekent. Hier kan momenteel alleen worden gekozen voor het LKB Model. In een volgende versie zal ook het TNO Bulk model [Kunz, 1995] worden geïmplementeerd (zie figuur 3.22).



Figuur 3.22: Micrometeorologisch model

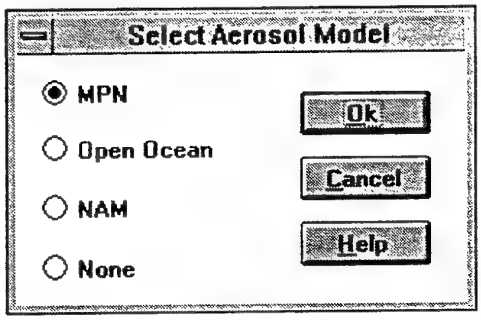
De volgende keuze betreft het Aerosol Model om de aerosol extinctie (α_A) op een hoogte van 10 meter te berekenen. Figuur 3.23 laat het selecteren van optie

Aerosol Model zien. Eveneens is in figuur 3.23 te zien dat op dit moment het MPN model staat ingesteld.

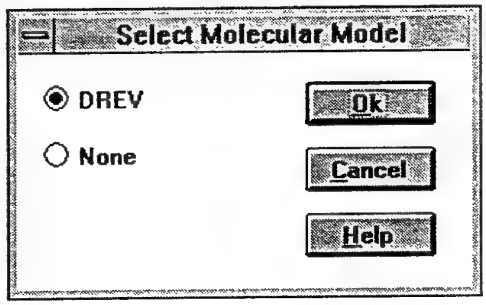


Figuur 3.23: Selectie Aerosol Model

Gekozen kan worden uit de modellen, weergegeven in figuur 3.24. De drie beschikbare modellen gelden elk voor een ander geografisch gebied (zie hoofdstuk 2).

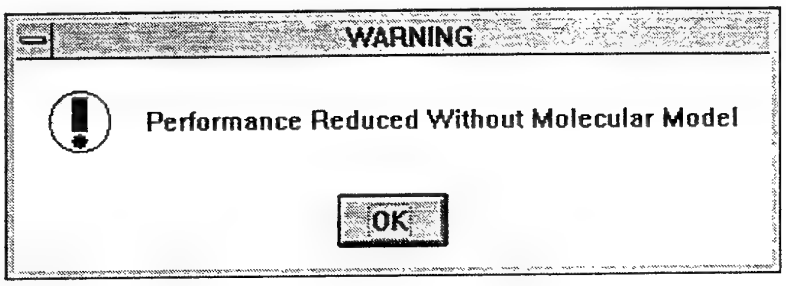


Figuur 3.24: Keuze modellen

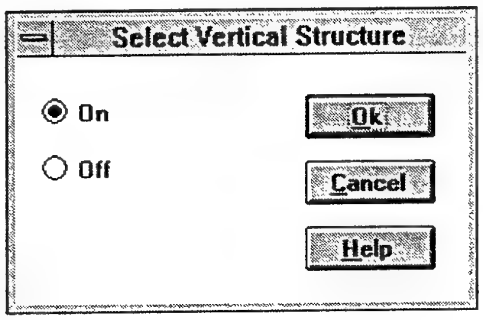


Figuur 3.25: Select Molecular Model dialogbox

In figuur 3.25 is de Select Molecular Model dialogbox weergegeven. Deze dialogbox dient om het model te selecteren dat de moleculaire extinctie berekent bij het maken van een berekening. De *default* selectie is het DREV model. Push-button Ok bevestigt de radio-button selectie en push-button Cancel annuleert de veranderingen. De Help optie binnen deze dialogbox is in deze versie van RAID niet geïmplementeerd.
Selectie *None* in de Select Molecular Model dialogbox resulteert in een messagebox zoals weergegeven in figuur 3.26. Deze *Warning* wijst erop dat de de reikwijdtevoorspeller zonder moleculair model minder nauwkeurig zal zijn.



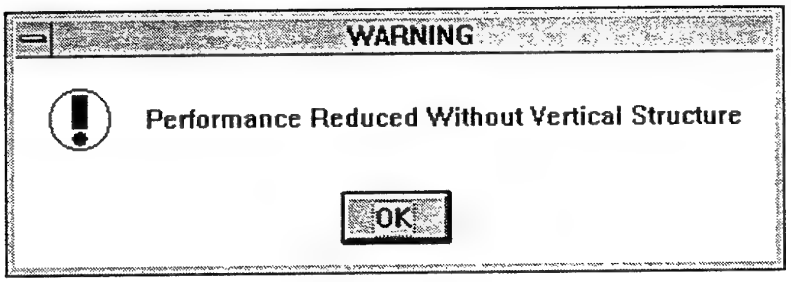
Figuur 3.26: Performance reduced



Figuur 3.27: Select Vertical Structure dialogbox

In figuur 3.27 is de Select Vertical Structure dialogbox weergegeven. De Select Vertical Structure dialogbox dient om de berekening van het verticale profiel van de extinctie aan of uit te zetten. Bij *off* worden alle berekeningen uitgevoerd voor een hoogte van 10 meter, bij *on* wordt het profiel van aerosol extinctie berekend met het NPS model, en het profiel van moleculaire extinctie met het DREV model. Push-button Ok bevestigt de radio-button selectie en push-button Cancel annuleert de veranderingen. De Help optie binnen deze dialogbox is in deze versie van RAID niet geïmplementeerd.

Selectie *None* in de Select Vertical Structure dialogbox resulteert in een message-box zoals weergegeven in figuur 3.28. Deze *Warning* wijst erop dat de *performance* zonder berekening van het verticale profiel zal worden gereduceerd.



Figuur 3.28: Performance reduced

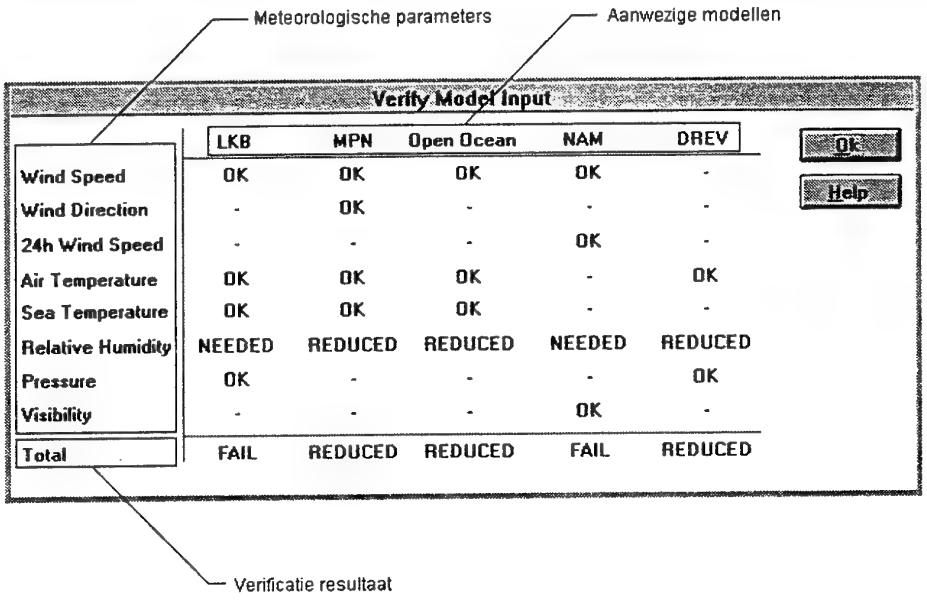
3.2.5 Plot

Nadat type berekening (Mode), invoerparameters (Input) en de te gebruiken modellen (Settings) ingevoerd zijn, kan met hoofdmenu optie *Plot* de berekening daadwerkelijk gestart worden. Het proces bestaat uit drie stappen:

- verificatie van de invoer
- berekening
- grafische presentatie van resultaten

Verificatie—Als eerste wordt een verificatie van de meteorologische parameters uitgevoerd. Als de verificatie zonder fouten is verlopen (dat wil dat zeggen dat de geselecteerde modellen voldoende parameters hebben) wordt de berekening gestart.

Het kan ook voorkomen dat één of meerdere modellen niet gebruikt kunnen worden omdat één of meerdere meteorologische parameters door middel van checkboxen in het input meteo dialogbox zijn uitgeschakeld. In dat geval wordt door middel van een tabel duidelijk gemaakt welke modellen niet gebruikt kunnen worden en welke modellen als alternatieven gekozen kunnen worden.



	LKB	MPN	Open Ocean	NAM	DREV
Wind Speed	OK	OK	OK	OK	-
Wind Direction	-	OK	-	-	-
24h Wind Speed	-	-	-	OK	-
Air Temperature	OK	OK	OK	-	OK
Sea Temperature	OK	OK	OK	-	-
Relative Humidity	NEEDED	REDUCED	REDUCED	NEEDED	REDUCED
Pressure	OK	-	-	-	OK
Visibility	-	-	-	OK	-
Total	FAIL	REDUCED	REDUCED	FAIL	REDUCED

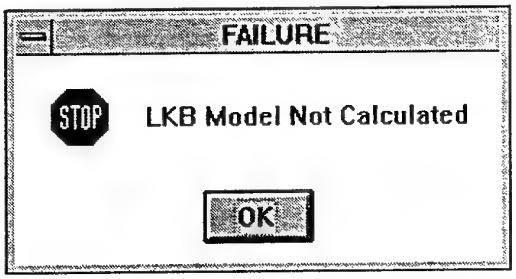
Figuur 3.29: Verify dialogbox

In figuur 3.29 wordt de Verify dialogbox getoond die verschijnt wanneer de fysische parameter *Relative Humidity* is uitgeschakeld. Een beschrijving van meldingen binnen deze dialogbox staat in tabel 3.1. De *Help* optie binnen deze dialogbox is in deze versie van RAID niet geïmplementeerd.

Tabel 3.1: Verify meldingen

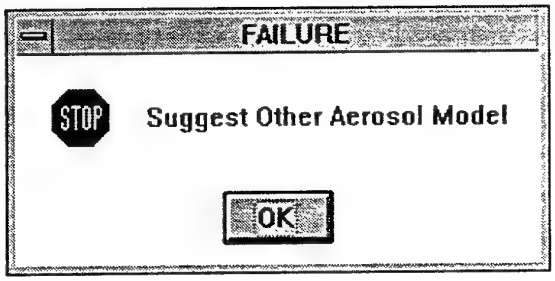
Melding	Betekenis
-	Parameter niet nodig voor betreffend model
FAIL	Betreffend model is niet bruikbaar door gebrek aan parameters
NEEDED	Parameter vereist om model te gebruiken
OK	Verificatie geslaagd
REDUCED	*Performance* van betreffend model wordt gereduceerd door gebrek aan parameter(s)

De Verify-dialogbox van figuur 3.29 laat zien dat zonder relatieve vochtigheid het LKB-model niet gebruikt kan worden. Zonder LKB-berekening is geen reikwijdte-voorspelling mogelijk. De gebruiker wordt hiervan op de hoogte gesteld door de messagebox in figuur 3.30

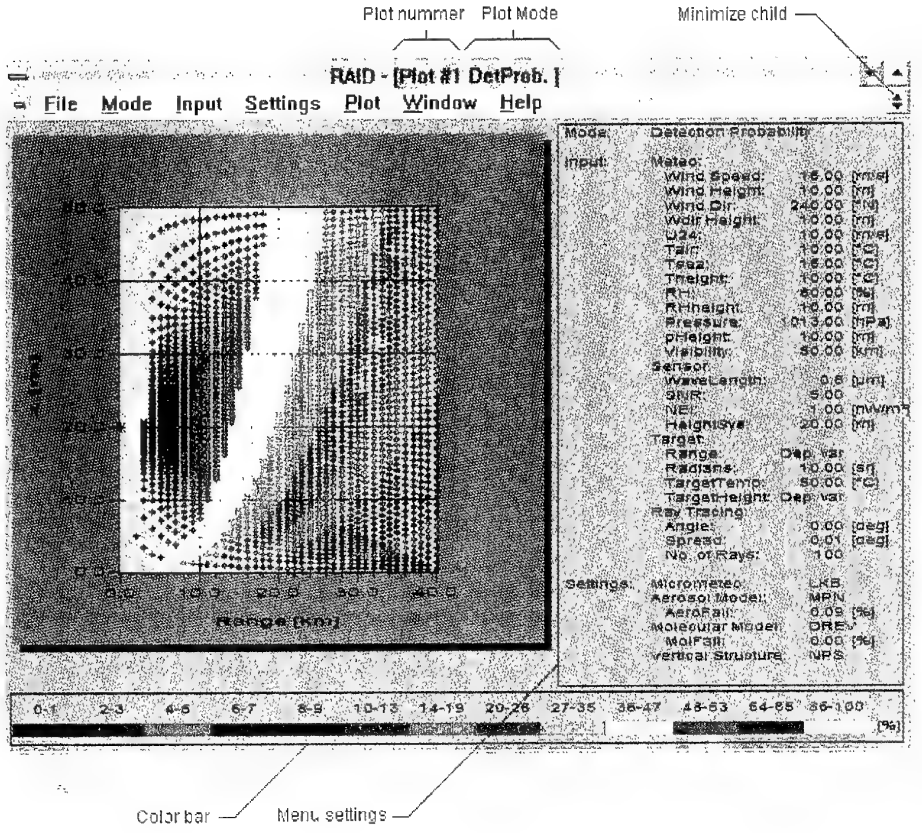


Figuur 3.30: LKB failure

In het geval van de berekening van de aerosol extinctie op 10 meter hoogte kan het nuttig zijn om een alternatief model te kiezen wanneer een bepaald model niet gebruikt kan worden door gebrek aan parameters. Hier volgt een voorbeeld: bij afwezigheid van 'wind richting' kan het MPN-model niet gebruikt worden. Als dit model geselecteerd was geeft RAID aan dat een *ander aerosol model* als alternatief gekozen kan worden (zie figuur 3.31).



Figuur 3.31: Aerosol model failure



Figuur 3.32: Plot window

Nadat de verificatie procedure is doorlopen wordt de eigenlijke berekening uitgevoerd (dit kan enkele minuten duren, waarbij de tekst *calculating* op het scherm getoond wordt). Na deze berekening wordt een plot window gegenereerd. Figuur 3.32 geeft aan uit welke onderdelen dit window is opgebouwd.

Plotnummer—Iedere plot heeft zijn eigen plotnummer. Zodra een plot-window wordt gesloten verdwijnt ook zijn plotnummer. Plotnummers van plots die zijn gesloten, kunnen weer worden gebruikt door nieuwe plots. Als bijvoorbeeld twee plots worden gemaakt en *Plot #1* wordt gesloten, dan krijgt de volgende plot die wordt gemaakt het nummer: *#1*.

Plot Mode—Net als het plotnummer verschijnt bij een plot de geselecteerde mode in de title bar van het window.

Minimize child—Door met de muis op deze button te klikken wordt het plot-window verkleind zodat dit komt te liggen binnen het window van RAID. Om het plot-window te verkleinen naar een icon, moet deze nogmaals verkleind worden door met de muis te klikken op de *minimize* button van dit window.
Om de plot weer zichtbaar te maken moet twee maal op de icon worden geklikt. Het gebruik van deze icons is handig om plots 'tijdelijk' op te bergen binnen RAID.

Iedere icon bevat het plotnummer, de plot Mode en geeft in het klein de betreffende plot weer (zie figuur 3.33).

Figuur 3.33: Plot icon

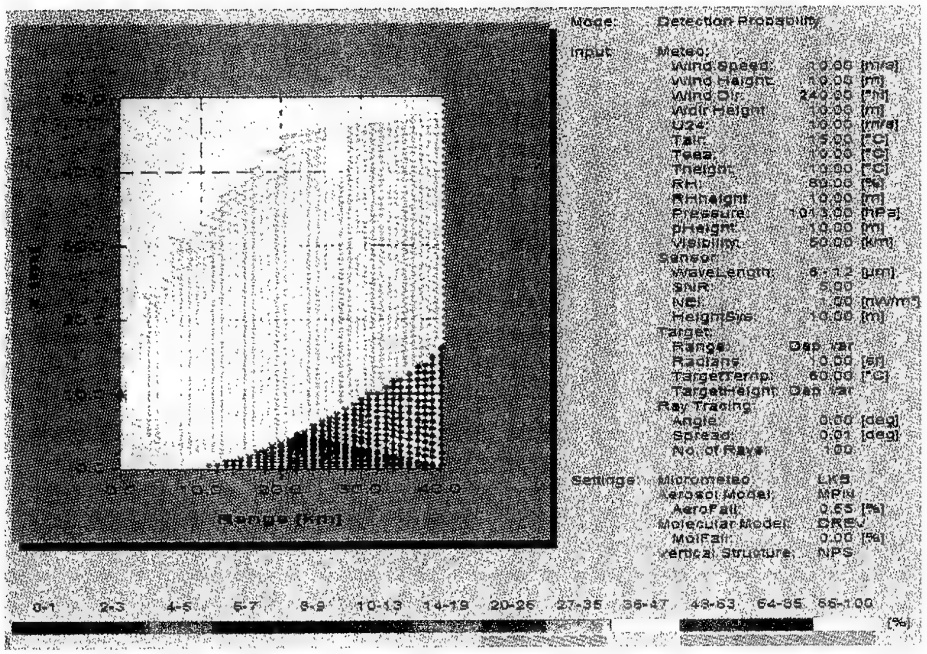
Color bar—Alleen in de plot Mode *Detection probability* wordt deze kleuren balk gebruikt. Deze kleuren balk geeft aan hoe de percentages zijn gekoppeld aan kleuren binnen de plot.

Menu settings—Binnen dit aangegeven kader bevinden zich relevante menu en dialogbox instellingen die betrekking hebben op de plot. Ook verschijnt de betrouwbaarheid (*AeroFail* en *MolFail*) van de gebruikte aerosol en moleculaire modellen naast de plot.

Zoals reeds opgemerkt in de paragraaf over hoofdmenu optie *Mode*, zijn er verschillende typen berekeningen mogelijk (bijvoorbeeld detection probability). Hieronder volgt een korte opsomming van de mogelijkheden

Mode: Detection Probability—Deze plot geeft de transmissie tussen doel en sensor weer als functie van de hoogte (z) en de afstand (*Range*). De ster op de verticale as geeft de hoogte aan, waarop de sensor is geplaatst. In deze plot is geen rekening gehouden met het verloop van lichtstralen door effecten van licht-breking tussen opeenvolgende luchtlagen.

Een Detection Probability plot die wordt verkregen door met de default settings een plot te genereren, wordt getoond in figuur 3.34. In deze plot is te zien dat met de ingestelde meteorologische parameters, de transmissie over een afstand tussen 0 en 30 km, tussen 86 en 100% ligt. Op een afstand tussen 30 en 40 km is deze transmissie afgenomen met 14% (gezien vanuit de sensor op 10 meter hoogte).

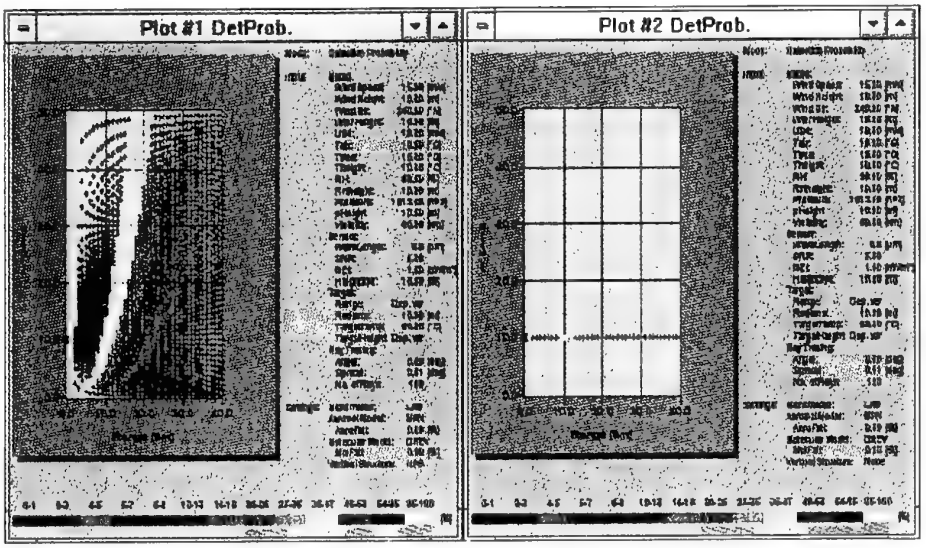


Figuur 3.34: Detection Probability

Zodra onder menu-optie *Settings* de verticale structuur berekening wordt uitgezet, zal de transmissie alleen worden gegeven als functie van de afstand (op een vaste hoogte van 10 meter). In figuur 3.35 is zowel een detection probability plot met, als een plot zonder verticaal profiel berekening te zien. De ingestelde parameters zijn in tabel 3.2 te zien.

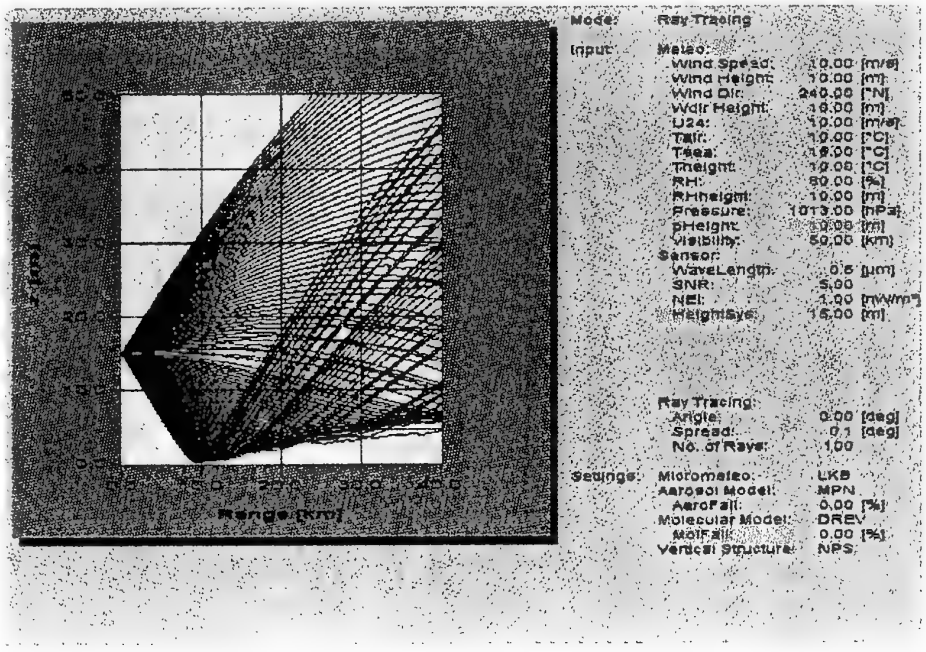
Tabel 3.2: Detection Probabitity

Menu optie	Parameter	Default	Ingesteld
Input-Meteo	Wind Speed	10.00 m/s	15.00 m/s
	Air Temperature	15.00 °C	10.00 °C
	Sea Temperature	10.00 °C	15.00 °C
Input-Sensor	Wavelength	8-12 µm	0.6 µm
	Sensor Height	10.00 m	25.00 m
Settings- Vertical Structure	NPS-Model / None	NPS-Model	NPS-model (links) None (rechts)



Figuur 3.35: Met / zonder verticale structuur

Mode: Ray Tracing—De Ray Tracing plot geeft de stralengang waarbij rekening wordt gehouden met refractieverschijnselen, dat wil zeggen de breking van licht ten gevolge van variaties in temperatuur en vochtgehalte als functie van de hoogte. Een voorbeeld van een dergelijke plot is te zien in figuur 3.36. In deze plot is tussen een range van 10.0 en 40.0 km, een gebied vlak boven het zeeoppervlak te zien, vanwaar geen stralengang naar de sensor mogelijk is. Dit betekent dat een doel in dit gebied niet zichtbaar is.



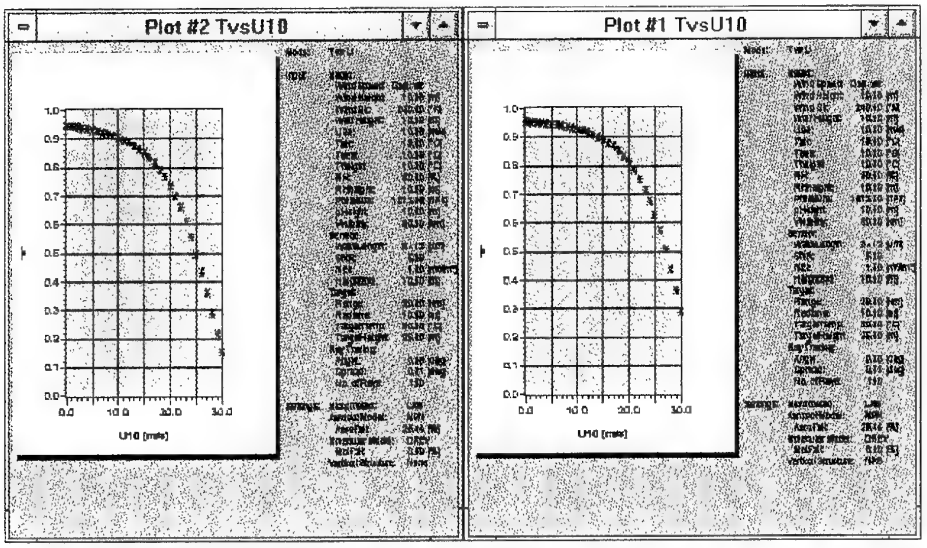
Figuur 3.36: Ray Tracing

Mode: T vs Wind Speed—De T vs Wind Speed plot geeft het verloop van de transmissie tussen doel en sensor als functie van de windsnelheid op 10 meter hoogte (U_{10}). De afhankelijke parameter van de plot (windsnelheid) wordt in de tekst naast de plot aangegeven als: Dep. var. (*dependent variable*). In figuur 3.37 is zowel een T vs Wind Speed plot zonder verticaal profiel te zien (links) als een dergelijke plot met verticaal profiel berekening (rechts).

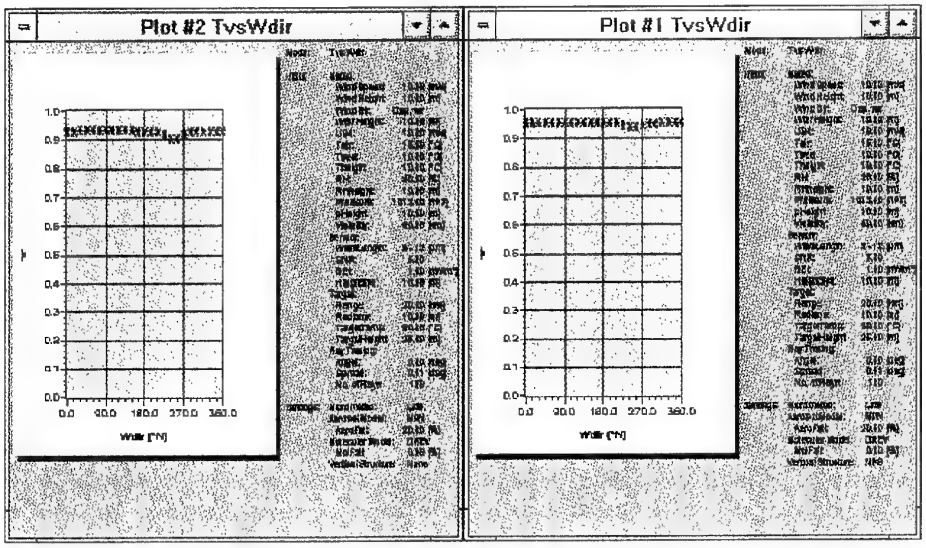
Mode: T vs Wind Direction—De T vs Wind Direction plot geeft het verloop van de transmissie door de atmosfeer weer als functie van de windrichting. De afhankelijke parameter van de plot (windrichting) wordt in de tekst naast de plot aangegeven als: Dep. var. (*dependent variable*). In figuur 3.38 is zowel een T vs Wind Direction plot zonder verticaal profiel te zien (links) als een dergelijke plot met verticaal profiel berekening (rechts).

Mode: T vs Relative Humidity—De T vs Relative Humidity plot geeft het verloop van de transmissie door de atmosfeer weer als functie van de relatieve vochtigheid. De afhankelijke parameter van de plot (relatieve luchtvochtigheid) wordt in de tekst naast de plot aangegeven als: Dep. var. (*dependent variable*).In figuur 3.39 is zowel een T vs Relative Humidity plot met verticaal profiel te zien (links) als een dergelijke plot zonder verticaal profiel berekening (rechts).

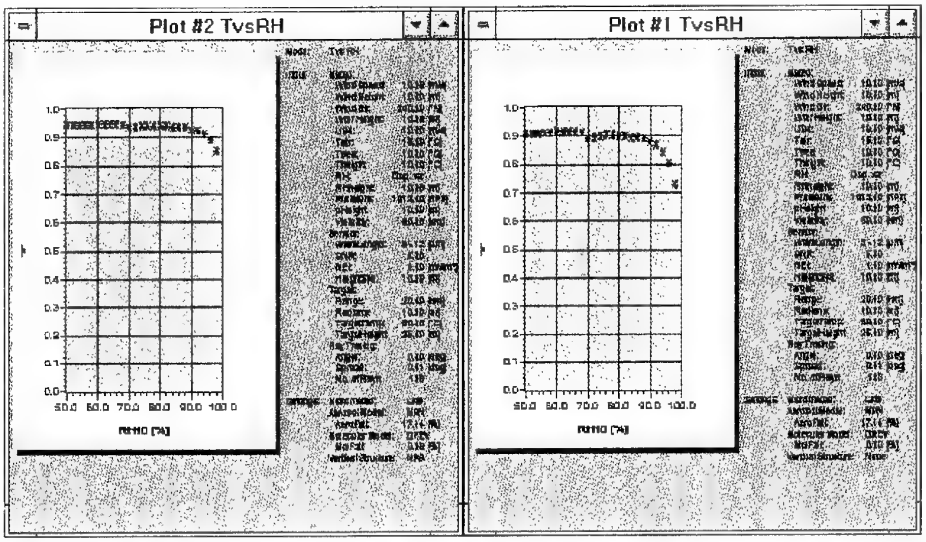
Mode: T vs ASTD—De T vs ASTD plot geeft het verloop van de transmissie door de atmosfeer weer als functie van de ASTD. De afhankelijke parameter van de plot (ASTD) wordt in de tekst naast de plot aangegeven als: Dep. var. (*dependent variable*). In figuur 3.40 is zowel een T vs ASTD plot met verticaal profiel te zien (links) als een dergelijke plot zonder verticaal profiel berekening (rechts).



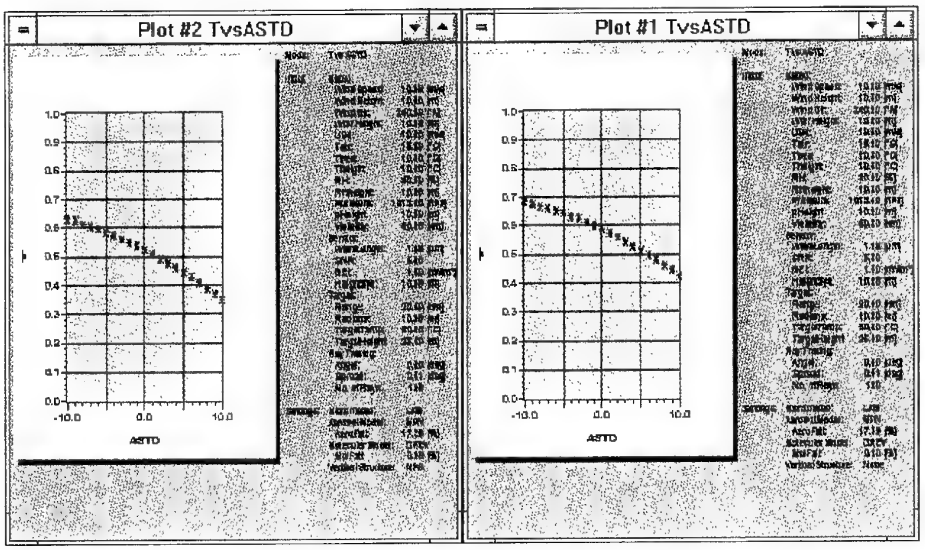
Figuur 3.37: T vs Wind Speed



Figuur 3.38: T vs Wind Direction

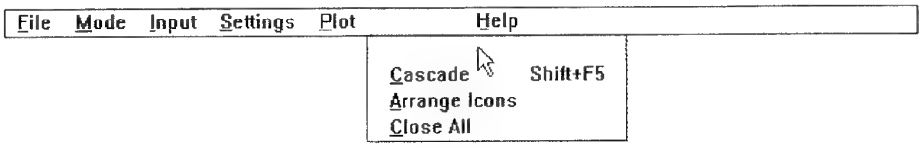


Figuur 3.39: T vs Relative Humidity



Figuur 3.40: T vs ASTD

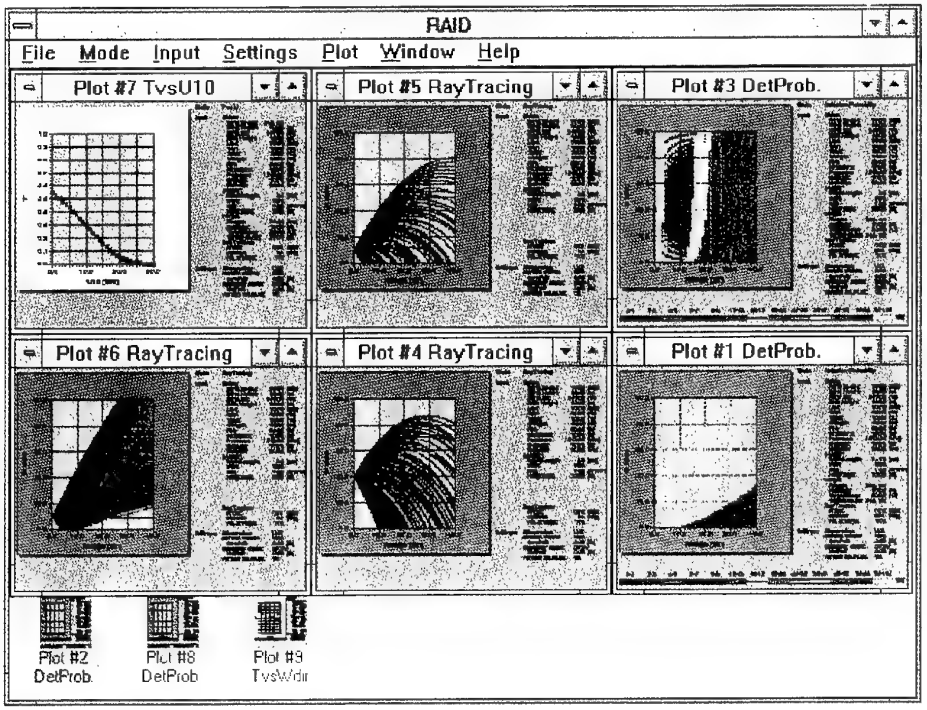
3.2.6 Window



Figuur 3.41: Window optie

Onder hoofdmenu optie *Window* (zie figuur 3.41) bevinden zich alle mogelijkheden voor het manipuleren van windows die gegenereerd zijn door RAID. Zodra menu optie *Plot* wordt geactiveerd wordt een window aangemaakt.

Als alle windows over elkaar heen geplaatst zijn, is het handig om gebruik te maken van de optie *Tile* (figuur 3.42) Tile zorgt voor het verkleinen van de windows op een evenredige manier en zet de windows naast elkaar op het scherm. Met Tile zijn dus altijd alle windows die niet minimized zijn tegelijk op het scherm te zien.



Figuur 3.42: Tile

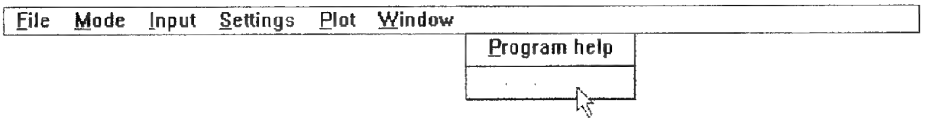
Cascade zorgt ook voor het weergeven van alle windows tegelijk, alleen worden nu de windows achter elkaar geplaatst op een zodanige manier dat het net lijkt of alle windows opgestapeld zijn.

De optie *Arrange Icons* kan handig zijn als een groot aantal plots gemaakt is en de icons van minimized windows door elkaar staan. Arrange Icons zet de icons in de juiste volgorde neer en plaatst de icons op vaste posities op het scherm.

De laatste optie is *Close All*. Deze optie sluit alle windows die er zijn. Ook minimized windows die als een icon binnen het MDI windows staan worden verwijderd. Deze optie bespaart tijd als veel plots gemaakt zijn.

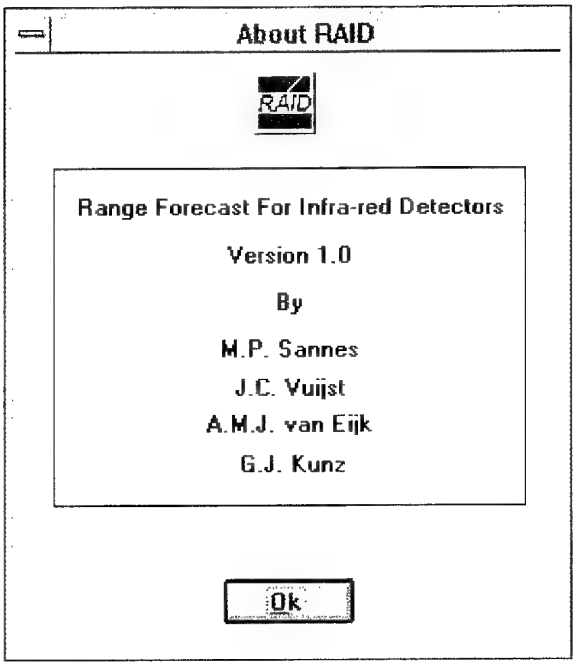
3.2.7 Help

Figuur 3.43 laat de optie *Help* zien. De help optie is nog niet geïmplementeerd. Delen uit deze handleiding zullen later als help pagina's worden toegevoegd binnen RAID. Help bevindt zich zoals de conventie (*Common User Access*, CUA) voorschrijft uiterst rechts op de menu balk.



Figuur 3.43: Help optie

Het selecteren van *About* zorgt voor de weergave van de dialogbox in figuur 3.44. About vertelt welk versie van het programma in gebruik is, en wie verantwoordelijk zijn voor het programma.



Figuur 3.44: Keuze About

3.3 Restricties

De reikwijdtevoorspeller kan niet in alle meteorologische omstandigheden gebruikt worden. In deze paragraaf worden de limieten van het model besproken.

Het LKB-model kan niet gebruikt worden in een zeer stabiele atmosfeer, d.w.z. voor een warme luchtmassa boven een koude zee ($\Delta T > 5$ °C) en een lage windsnelheid (< 5 m/s). In deze situaties worden de verticale profielen van windsnelheid, vochtigheid en temperatuur niet correct uitgerekend, hetgeen leidt tot afwijkingen in de voorspelde extinctie. Daarnaast geldt, ongeacht de stabiliteit van de atmosfeer, dat de theorie die ten grondslag ligt aan het LKB-model alleen geldt voor de onderste laag van de atmosfeer ($z < 100$ m).

Het moleculaire extinctie model geeft in principe een correct resultaat voor alle meteorologische scenario's die met de reikwijdtevoorspeller worden geëvalueerd. Wel dient de luchttemperatuur zo nauwkeurig mogelijk te worden gespecificeerd.

De aerosol extinctie kan met drie verschillende modellen worden berekend. Indien alle parameters beschikbaar zijn, geeft het NAM model een redelijke voorspelling voor open oceaan condities.

Het TNO Open Oceaan model kan gebruikt worden voor de Noord-Atlantische oceaan voor windsnelheden kleiner dan 15 m/s. De betrouwbaarheid neemt af als de atmosfeer instabiel is (ASTD < -5 °C) of de lucht warm is (> 15 °C). De extinctie in de 3-5 μm band wordt minder goed berekend naarmate het zicht minder wordt (< 1 km); de extinctie in de 8-12 μm bandextinctie wordt minder betrouwbaar bij lage windsnelheden (< 5 m/s) en/of hoge relatieve vochtigheden (> 90%).

Het MPN model is bruikbaar in een straal van 40 km rondom het MPN platform. . De betrouwbaarheid van de voorspelling neemt af wanneer de windrichting tussen 0° en 90° N ligt en/of wanneer de luchttemperatuur hoger is dan 20 °C. Verder gelden dezelfde restricties voor de 3-5 en 8-12 μm banden als genoemd bij het Open Ocean model.

Het model dat het verticale profiel van de aerosolextinctie uitrekent is minder betrouwbaar in kustgebieden met aflandige wind.

4. Technische informatie

4.1 Programmeeromgeving

4.1.1 Het OOP concept

Om in het kort te vertellen wat OOP (Object Oriented Programming) is, is het noodzakelijk te weten hoe OOP precies is ontstaan. Hiervoor even een korte terugblik in de geschiedenis van de computer industrie.

In het begin van het computertijdperk moesten computerprogrammeurs hun machines aan het werk zetten door programma's en gegevens min of meer rechtstreeks in binaire code aan de machine door te geven. Soms ging dat met schakelaars voor adressen en instructies. Deze binaire instructies worden machinetaal genoemd. Daar dit al snel een tijdrovende bezigheid werd, besloot men voor de instructies afkortingen te gebruiken, zogenaamde *mnemonics*. Dit werd assembler genoemd. Beide vormen van programmeren vereisen een gedetailleerde kennis van een specifiek computersysteem. Aan het oplossen van een probleem met de computer kwam men nauwelijks toe.

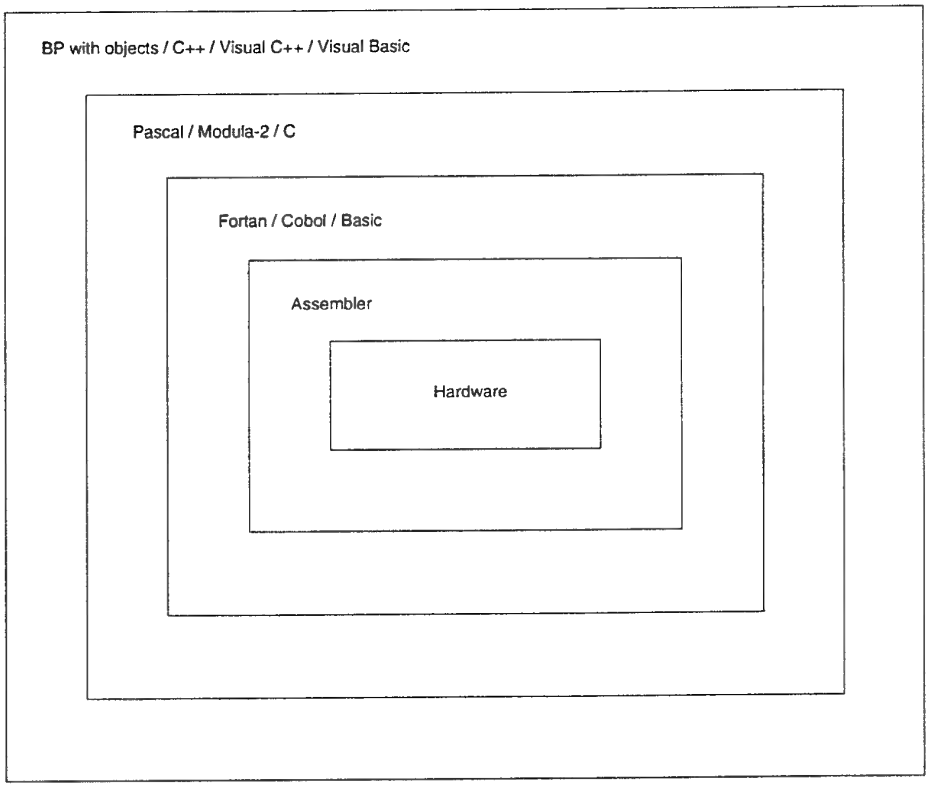
Vandaar dat eind jaren vijftig gezocht werd naar talen die op een computer geïmplementeerd konden worden. Deze talen vormden een scheiding tussen de architectuur van de machine en de buitenwereld. Het probleem moest dus geformuleerd worden binnen de programmeertaal. Zo ontstonden Fortran, Cobol en Basic.

Programmeurs waren nu verlost van het probleem dat specifieke details van het computersysteem bekend moesten zijn. Wel dienden zij het probleem in de beschikbare commando's van de programmeertaal te formuleren. Om te voorkomen dat telkens weer het wiel opnieuw werd uitgevonden, werden programmabibliotheken aangemaakt. Maar de programmatuur moest dan wel precies zijn wat men zocht.

Het duurde niet erg lang of de omvang van de aan te pakken problemen werd zo groot en zo complex dat men naar wegen ging zoeken om het probleem te analyseren. Na de analyse kon met een redelijke kans op succes het probleem door programmeurs in de machine worden ingebracht. Op deze manier is gestructureerd programmeren onstaan. Dit leidde tot talen als Pascal, Modula-2 en C.

Begin jaren tachtig zijn de methodieken gestructureerd ontwerpen en gestructureerd analyseren (*Structured Analysis and Structured Design*) ontwikkeld. Deze technieken zijn de logische voortzetting van gestructureerd programmeren, doordat zij het hele proces van programma ontwikkeling (analyse, ontwerp, implementatie, testen en onderhoud) beschouwen. Ofschoon ook de gegevens-beschrijvingen beter werden, werd aan de relatie tussen gegevens en bewerkingen nog niet veel gedaan.

Figuur 4.1 toont de onwikkeling van de programmeertalen van lage taal (assembler) naar hogere taal (hoger betekent meer detail afscherming van de hardware).



Figuur 4.1: Ontwikkeling programmeertalen

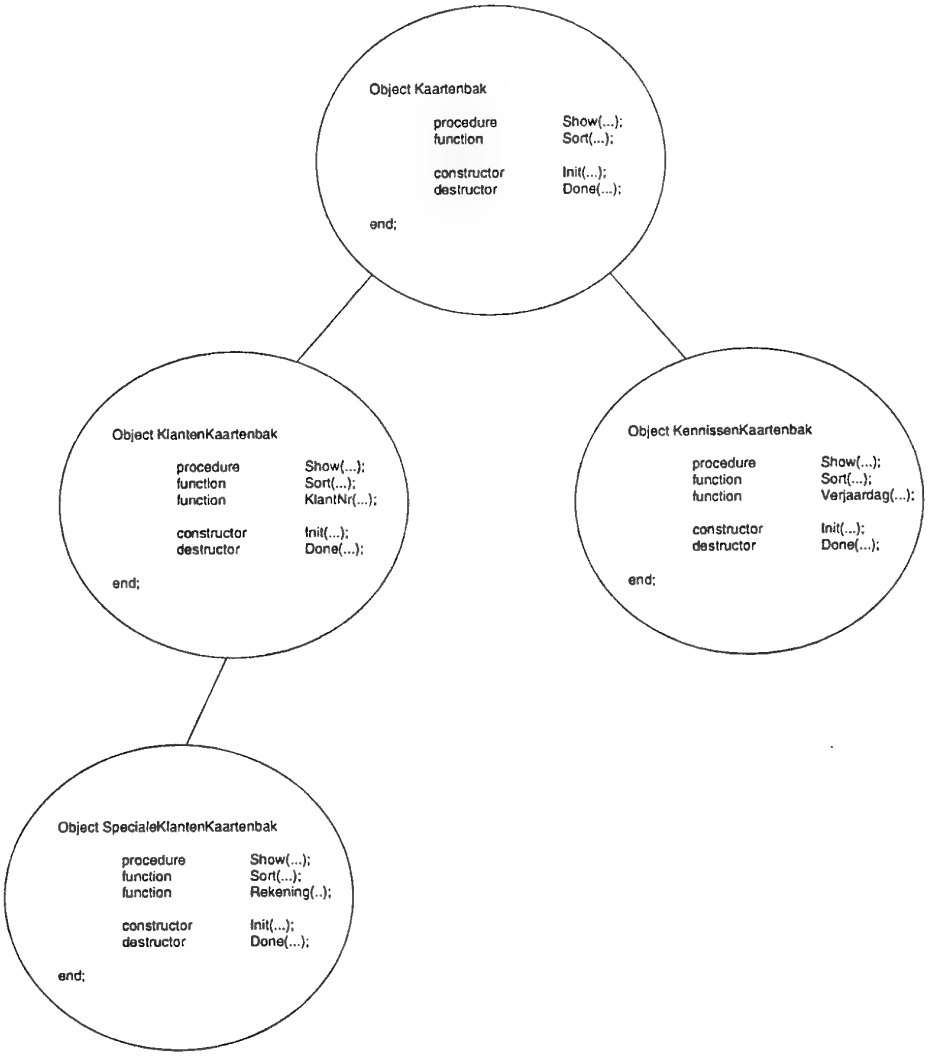
De laatste jaren is een belangrijk nieuw concept ontwikkeld. In plaats van gegevens en bewerkingen als separate zaken te beschouwen, worden deze als één geheel benaderd, namelijk als zogenaamde objecten. Ieder object kent zijn eigen verzameling handelingen.

Een taal die gebaseerd is op het gebruik van objecten, noemt men een object georiënteerde taal en het programmeren in een dergelijke taal wordt aangeduid met *Object Oriented Programming* (OOP).

De drie grondslagen waar het *object oriented* principe op berust zijn:

- *Encapsulation* (Inkapselen): het samenvoegen van een record en de procedures en functies die het record manipuleren.
- *Inheritance* (Erven): een object definiëren en dan gebruiken om een aantal onderliggende objecten in de hiërarchie te beschrijven. Deze objecten maken dan gebruik van alle code en data uit de bovenliggende objecten in de hiërarchie.

- *Polymorphism* (Meerdere verschijningen): een message die gebruikt kan worden naar boven en beneden in de hiërarchie. Ieder object in de hiërarchie implementeert de actie op de manier zoals het bij hemzelf past.



Figuur 4.2: Objecten voorbeeld

In figuur 4.2 worden deze principes aan de hand van een voorbeeld geïllustreerd. Alle objecten hebben een procedure Show, die betrekking heeft op het object zelf (polymorphisme). Het is dus mogelijk om Show tegen meerdere objecten 'te zeggen' (evenals Init, Done en Sort). De objecten maken alle gebruik van *encapsulation*: procedures en functies (maar ook variabelen, typen en constanten) behoren tot het object zelf. *Inheritance* is te zien bij het object SpecialeKlantenKaartenbak dat zich onder het object KlantenKaartenbak bevindt.
Binnen het object SpecialeKlantenKaartenbak zijn alle procedures en functies (en variabelen etc.) van het object KlantenKaartenbak bereikbaar. Maar ook de procedure en functies (en variabelen etc.) van het object Kaartenbak zijn bereikbaar.

De voornaamste reden voor het gebruik van object georiënteerd programmeren is dat het voordeel biedt bij het opbouwen van een (groot) programma. De traditionele procedurele aanpak zorgt ervoor dat specificaties (datavelden, procedureheaders etc.) nauwkeurig van tevoren bekend moeten zijn, terwijl dat bij een object geörienteerd programma niet direkt het geval hoeft te zijn.
Wel zal een minimale basis (de object structuur) aanwezig moeten zijn. Van hieruit kunnen de eigenschappen van objecten en hun eigen interne gedragingen (methods) ontworpen worden. Deze gedragingen kunnen tijdens het opbouwen van het uiteindelijke programma makkelijk worden gewijzigd, hetgeen dat bij een procedurele aanpak veel meer inspanning zou kosten.
Om de voordelen van object georiënteerd programmeren ten volle te benutten, dient het programma op een andere manier ontwikkeld te worden. Een methode die veel gebruikt wordt, is *incremental delivery approach*. Hierbij geldt dat het basisprogramma slecht éénmaal wordt gemaakt en dat dit vervolgens herhaaldelijk wordt aangepast en uitgebreid.
De tegenwoordig meest verbreidde *object oriented* talen zijn C++, Smalltalk en Borland Pascal for Windows. Er zijn zelfs *object oriented* talen ontwikkeld die in een grafische omgeving objecten 'samenvoegen' (Visual C++ en Visual Basic). Hier genereert de compiler zelf de benodigde code voor koppeling van objecten. Dit grafisch ontwerpen van een programma zal in de komende jaren steeds meer toegepast worden. De onderliggende code bestaat dan geheel uit een object georienteerde structuur. De procedurele structuur zal daardoor langzaam maar zeker verdwijnen.

4.1.2 Borland Pascal for Windows

De firma Borland levert het pakket Borland Pascal for Windows. Het pakket is bedoeld voor alle gebruikers die applicaties willen ontwikkelen voor een DOS en/of Windows[1] platform.

Het is mogelijk om voor drie soorten platforms, ook wel *targets* genoemd, een applicatie te schrijven. Deze zijn :

- *DOS real-mode target*: hiervoor worden applicaties geschreven die alleen geschikt zijn voor DOS in de eenvoudigste mode (16-bits). In de toekomst zal deze mode steeds minder gebruikt gaan worden.
- *Windows target*: hiervoor worden Windows applicaties geschreven, die niet meer te gebruiken zijn in DOS.
- *DOS protected mode target*: hiervoor worden DOS applicaties geschreven die maximaal gebruik maken van de processor (32-bits). Aangezien Windows bij 386 en snellere processors al in de *protected* mode werkt, is de DOS *protected mode* in de toekomst overbodig.

---

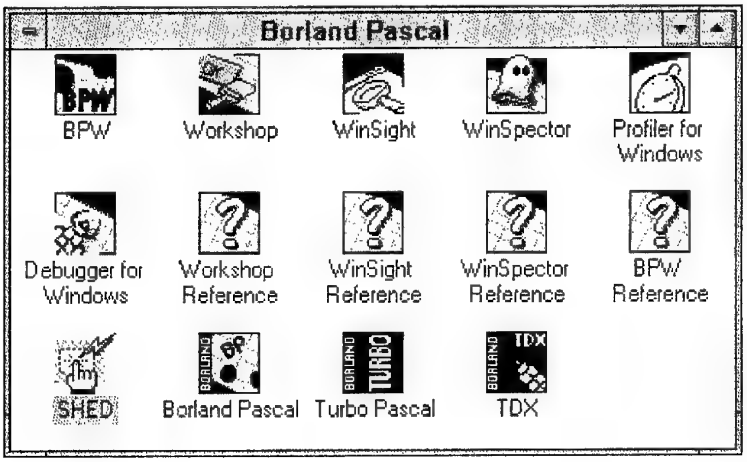
1 Windows met een hoofdletter refereert naar de applicatie Microsoft Windows. Wanneer verwezen wordt naar een window binnen deze applicatie, wordt een kleine letter gebruikt.

Bij het schrijven van een programma wordt gebruik gemaakt van een geïntegreerde ontwikkel omgeving (IDE, *Integrated Development Environment*) die bij Borland Pascal zowel in Windows als in DOS aanwezig is. De IDE bevat een editor met ingebouwde compiler. Ook kunnen tools als Turbo debugger en de resource workshop kunnen binnen de IDE ingelezen worden. Het is mogelijk om een DOS applicatie in de Windows IDE te schrijven. Hierdoor kan verwarring ontstaan en is het dus raadzaam om DOS en Windows programma's gescheiden te houden.

Nadat Borland Pascal for Windows geïnstalleerd is, wordt in windows een nieuwe groep aangemaakt. Deze groep heeft default de naam (figuur 4.3):

Figuur 4.3: Default naam

In figuur 4.4 is te zien welke programma's zich binnen deze groep bevinden.



Figuur 4.4: Programma's binnen groep

In figuur 4.4 staan alle programma's die bij het pakket horen.
Hier volgt een korte beschrijving van de belangrijkste programma's die betrekking hebben op het maken van een applicatie voor een Windows *target*:

- *Debugger for Windows*: een hulpprogramma voor het onderzoeken van een eigen geschreven applicatie, om zo mogelijke fouten op te sporen.
- *BPW*: Borland Pascal for Windows. De IDE van Borland Pascal.
- *Workshop*: Een programma om besturings elementen (resources) te creëren (bijvoorbeeld een dialogbox, drukknop, bitmap etc). Deze elementen kunnen dan in een eigen applicatie ingelezen worden.

- *WinSight*: voor het analyseren van alles wat met windows gebeurt. De zogenaamde *messages* die van en naar windows gaan alsmede informatie over window classes en de windows zelf.
- *WinSpector*: voor het analyseren van windows voordat een *crash* is opgetreden (UEA, Unrecoverable Application Error). Zo kan gezien worden waardoor de *crash* ontstond.
- *Profiler for Windows*: voor het bekijken van de snelheid van programma's.

4.1.3 OOP binnen Borland Pascal for Windows

Borland heeft een extensie gemaakt op de taal Turbo Pascal, die ervoor zorgt dat gewoon Turbo Pascal, zonder objecten, gecombineerd kan worden met objecten. Gesproken wordt dan van een hybride taal (Borland Pascal for Windows).

Doordat Borland Pascal een hybride taal is, zijn veel gebruikers het *object oriented* principe als een handige tool gaan zien dat makkelijk te combineren is met 'gewone' Pascal statements. Op deze manier is grote verwarring ontstaan over het nut van OOP. Vergeten wordt dat OOP een andere manier van programmeren is waarbij anders tegen een probleem moet worden aangekeken en waarvan de oplossing anders geïmplementeerd moet worden. Helaas heeft Borland hier zelf aan bijgedragen door een aantal wezenlijke OOP gedachten niet, of zeer onduidelijk, in de handleiding te beschrijven. Een duidelijk voorbeeld zijn de zogenaamde constructoren en destructor methods (methods zijn de procedures en functies binnen een object). Hieronder wordt dieper ingegaan op virtual methods, constructors en destructors.

Binnen het *object oriented* principe worden *virtual methods* gebruikt, die het *inheritance* principe tot uitdrukking brengen. De benaming *virtual* wordt hier gebruikt in de betekenis van: aanwezig, maar niet werkzaam. Het volgende voorbeeld kan dit verduidelijken:

Stel het volgende objecttype voor:

```
Rechthoek = object
   Coordinaten : TCoordinaten
   procedure VraagPlaats   (LBovenhoek : TCPaar);
   procedure VraagRHAttr (Lengte, Breedte :word);
   procedure TekenRechthoek;
end {Rechthoek};
```

Procedure Vraagplaats vraagt de gebruiker om coördinaten via het toetsenbord in te voeren. Ook is bekend dat de method TekenRechthoek de andere twee methods (VraagPlaats en VraagRHAttr) zal aanroepen. Wanneer nu gevraagd wordt de coördinaten vanuit een bestand te lezen, is de volgende uitbreiding noodzakelijk:

```
AndereRechthoek = object (Rechthoek)
   procedure VraagPlaats (LBovenhoek : TCPaar);
end {AndereRechthoek};
```

In het nieuwe object AndereRechthoek wordt een nieuwe method VraagPlaats gedefiniëerd die de coördinaten uit het bestand leest. Als *descendant* van Rechthoek erft het object AndereRechthoek de *methods* VraagRHAttr en TekenRechtHoek (maar niet VraagPlaats!). Het is daarom mogelijk binnen het object AndereRechthoek de *method* TekenRechthoek aan te roepen. Zal bij deze aanroep dan ook de rechthoek op het scherm verschijnen met de coördinaten die vanuit het bestand worden gelezen? Het antwoord hierop is: Nee!

De *method* TekenRechthoek blijft via het toetsenbord om invoer vragen. De oorzaak hiervan is de werking van de compiler. Wanneer de compiler de aanroep van TekenRechthoek tegenkomt, levert dit op zich geen problemen op. Weliswaar kent het object AndereRechthoek de method TekenRechthoek niet, maar de compiler ziet wel dat AndereRechthoek een descendant van Rechthoek is. De compiler gaat vervolgens de method TekenRechthoek in het object Rechthoek zoeken.

Bij compilatie van method TekenRechthoek komt de compiler derhalve een aanroep naar Vraagplaats tegen. De compiler kijkt binnen het huidige object (Rechthoek!) om zich heen en vindt daar de *method* Vraagplaats. Alleen, dit is niet de Vraagplaats die de gebruiker bedoeld had.

Het probleem is dat de compiler moet begrijpen welk object de *method* TekenRechthoek aangeroepen heeft. Er moet pas een koppeling van de aangeroepen *methods* plaatsvinden zodra duidelijk is welk object de method gebruikt. Feitelijk is dat pas bekend zodra het programma uitgevoerd wordt.

In Borland Pascal zijn 2 soorten koppelingen (Engels: *binding*) gedefiniëerd:

- Statische binding: binding tijdens de compilatie (compile)
- Dynamische binding: binding wanneer method wordt aangeroepen (*runtime*)

In ons voorbeeld is een dynamische binding nodig. Dit kan worden geforceerd door achter de method het directive **virtual** te schrijven. Dit directive zorgt ervoor dat tijdens compilatie een virtuele ("aanwezig, maar niet werkzaam") verbinding tot stand wordt gebracht. De code wordt daarmee:

```
Rechthoek = object
   Coordinaten : TCoordinaten
   procedure VraagPlaats (LBovenhoek : TCPaar); virtual ;
   procedure VraagRHAttr (Lengte, Breedte :word);
   procedure TekenRechthoek;
end {Rechthoek};
```

```
AndereRechthoek = object (Rechthoek)
   procedure VraagPlaats (LBovenhoek : TCPaar); virtual ;
end {AndereRechthoek};
```

Het directive **virtual** wordt bij alle *descendants* aangebracht én bij het betreffende object zelf. Wordt dit niet gedaan, dan genereert de compiler een foutmelding en vraagt een andere naam voor de *method* in de descendant, omdat hij deze naam al kent in het object.

Bij een lange lijn van *descendants* wordt het tijdrovend om tijdens runtime deze hele lijst af te lopen om te vinden welke *method* op dat moment bij welk object hoort. Daarom is de **constructor** ontwikkeld. Ieder objecttype dat virtuele *methods* kent, moet een constructor hebben. Meestal wordt de naam Init gekozen voor de constructor *method*, maar er kan net zo goed een andere naam gebruikt worden. Zodra de compiler een objecttype tegenkomt met daarin een constructor, wordt voor dat objecttype een tabel (*Virtual Method Table*, VMT) in het datasegment aangemaakt. Het object krijgt een pointer naar deze tabel en in de tabel worden, naast wat eigenschappen over het object, pointers naar de code van de *methods* geplaatst. Vervolgens kan tijdens runtime de gewenste methode via het omleggen van pointers gevonden worden.

De constructor wordt aangeroepen (door de *runtime* compiler) wanneer er een instance van een object nodig is, dat wil zeggen wanneer een var declaratie wordt gedaan. De overstap van een abstractie (type declaratie) naar een vorm waarbij het object echt in het geheugen bestaat (var declaratie) noemt men het creëren van een instance. Om een *instance* van een object te maken wordt een uitgebreidere versie van de Pascal functie New() gebruikt. Het algemene formaat van deze New functie luidt:

```
function New (ObjectPointer, <constructor naam> ): Pointer;
```

De totale programma opzet ziet er nu zo uit:

```
type
Coordinaten = record
   X1, Y1, X2, Y2 : word;
end { Coordinaten };

Rechthoek = object
   Coordinaten : TCoordinaten
   constructor Init
   procedure VraagPlaats (LBovenhoek : TCPaar); virtual ;
   procedure VraagRHAttr (Lengte, Breedte :word);
   procedure TekenRechthoek;
end {Rechthoek};
```

```
AndereRechthoek = object (Rechthoek)
   constructor Init (Naam_Bestand: PChar);
   procedure VraagPlaats (LBovenhoek : TCPaar); virtual ;
end {AndereRechthoek};

{ Hier de code voor implementatie van de objecttypen }

var  ARH: AndereRechtHoek;
begin
   New (ARH, Init ('c:\data\coord.dat'));
   .....
   .....
end.
```

Naarmate meer *instances* gecreëerd worden, neemt de beschikbare ruimte op de heap af. Daarom is de **destructor** ontwikkeld als tegenpool van de constructor. Deze procedure (meestal Done genoemd) geeft alle geheugen vrij dat voor de betreffende *instance* van het object in beslag wordt genomen. Het gebruik van een destructor is niet verplicht. In zijn eenvoudigste vorm is de bijbehorende code:

```
destructor Done;
begin
end;
```

4.1.4 Het gebruik van MDI

MDI staat voor *multiple document interface*. Deze interface is een standaard voor Windows applicaties en stelt de gebruiker in staat om meerdere dokumenten gelijktijdig geopend te hebben. Dit kunnen Pascal *files* zijn, maar het is net zo goed mogelijk om bijvoorbeeld een spreadsheet file gelijktijdig open te hebben. De MDI standaard is onderdeel van de *Common User Access* specificatie van IBM. Voor de *Common User Access* specificatie wordt verwezen naar paragraaf 4.2.4.

ObjectWindows is het onderdeel van Borland Pascal for Windows dat zorgt voor alle aansturingen van windows. ObjectWindows bevat onder andere objecten om zelf een MDI applicatie te creëren. Iedere MDI applicatie heeft een aantal vaste componenten. Deze componenten bestaan uit:

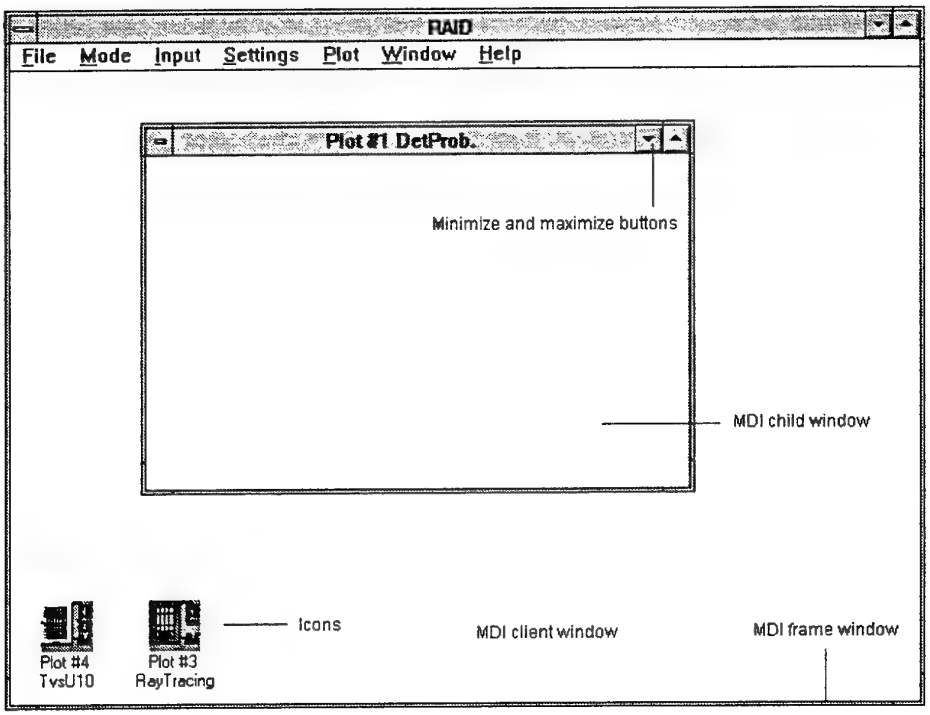
- een *main* window (frame window)
- een onzichtbaar window (client window)
- één of meerdere child windows

Binnen de *client area*[2] (het oppervlak binnen een window) van het *frame* window bevindt zich een onzichtbaar window. Dit window wordt het MDI *client* window

---

2 De client area van een window omvat het gebied onder de menu-bar, binnen het frame van een window. Dit gebied wordt ook wel de 'work' area genoemd.

genoemd. Dit *client* window bevat MDI *child* windows. In onderstaande figuur 5 zijn alle kenmerken van een MDI applicatie te vinden.



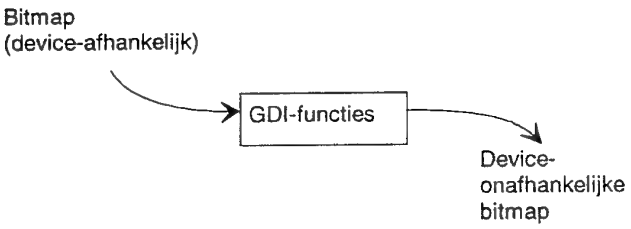
Figuur 4.5: MDI Applicatie

4.1.5 Het werken met bitmaps

Een bitmap is de actuele inhoud van een stuk geheugen dat de display context[3] voor een bepaald device bevat.
Daarom zijn bitmaps afhankelijk van het soort device dat moet worden aangesproken. Doordat devices niet noodzakelijkerwijs compatible met elkaar zijn kan deze device-afhankelijkheid een probleem worden. Het GDI (Graphics Device Inter-

[3] Device en display contexts—Bij DOS applicaties is het mogelijk om direct pixels te schrijven naar een beeldscherm of printer. Bij Windows applicaties moet dit via een device context gebeuren. Een device context is een 'virtueel device' waar alle grafische functies naar toe worden gestuurd. Aan elke device context is een fysiek device gekoppeld. De omzetting van het virtuele device naar het fysieke device wordt verzorgd door de device-driver van het fysieke device. Een device context bevat een aantal drawing tools zoals pens, brushes fonts, background colors, text colors en current positions. Voor een applicatie zien alle device contexten er hetzelfde uit, dus onafhankelijk van het soort device. In tegenstelling tot het hele window (frame + menu + client area), representeert een device context alleen het client area. Windows levert speciale device contexts voor client areas van windows. Deze device contexts worden display contexts genoemd. Met deze display context hoeft de programmeur zich geen zorgen te maken over offsets ten gevolge van de positie van het window.

face)[4] levert onder andere een aantal technieken om deze problemen te verhelpen door middel van zogenaamde *device independent* bitmaps (zie figuur 4.6).



Figuur4.6: GDI-functies

GDI-functies die bitmaps creëren zijn:
- CreateCompatibleDC
- CreateCompatibleBitmap
- CreateDIBitmap

GDI-functies die bitmaps manipuleren zijn:
- BitBlt
- StretchBlt
- StretchDIBits
- SetDIBitsToDevice

In de praktijk blijkt dat het GDI onvoldoende functies bevat om alle grafische afbeeldingen te beschrijven. In dat geval kan het raadzaam zijn device-afhankelijke bitmaps te gebruiken.
Voor het weergeven van de resultaten van RAID is gekozen voor bitmaps. Deze bitmaps zijn onafhankelijk van het display-type. De GDI-functie die dit mogelijk maakt is: CreateCompatibleBitmap. Door het gebruik van bitmaps dient de berekening slechts éénmaal te worden uitgevoerd. Vanwege de benodigde rekentijd voor de diverse modellen leidt dit tot een aanzienlijke tijdbesparing.
Dit komt tot uiting in:

- De mogelijkheid om windows snel te rangschikken met menu opties (Window-) Tile en (Window-) Cascade.
- De mogelijkheid om een plot met behulp van GDI-functies snel om te zetten naar elke gewenste grootte, zodat met de beschikbare *resize*-functies de plot kan worden vergroot of verkleind. De plot kan bijvoorbeeld worden vergroot

---

4 GDI functies geven de mogelijkheid om binnen een Windows-applicatie grafische afbeeldingen te manipuleren en/of af te beelden. GDI geeft een device (bijvoorbeeld VGA display of postscript printer) grafische mogelijkheden, onafhankelijk van het soort device dat wordt gebruikt. Het is dus mogelijk om dezelfde functies te gebruiken voor zowel het aansturen van een monitor als het aansturen van een postscript printer. GDI bereikt deze device-onafhankelijkheid door gebruik te maken van device drivers die GDI functies omzetten in commando's voor het betreffende device.

naar de grootte van het client-area van het window, maar kan ook worden verkleind naar een icon.

Het gebruik van bitmaps heeft echter ook enkele nadelen. In het algemeen is een bitmap een vrij inefficiënte manier om een afbeelding te beschrijven, omdat iedere *pixel* met een eigen kleur aangegeven is. Dit kost veel geheugen. Ook zal bij het verkleinen van een plot resolutie-vermindering optreden. Dit komt doordat de *pixels* niet meer gelijk verdeeld komen te liggen door de beperkte ruimte binnen het verkleinde window. Daarom is het raadzaam om bij het analyseren van een plot, deze met de *maximize* button te vergroten naar de grootte van het *client-area* van het MDI *frame* window.

4.1.6 Transfer buffers

Wanneer een dialogbox of window met *controls* (bijvoorbeeld *checkboxes* of *editcontrols*) tijdens *runtime* meerdere malen geopend en gesloten wordt, is het (meestal) gewenst dat het window zich, bij het opnieuw openen, in dezelfde staat bevindt als toen het werd gesloten. Dit kan gebeuren met behulp van *transfer buffers*. Een *transfer buffer* is een record dat de staat van alle *controls* van een window of dialogbox bewaart. De applicatie zelf kan het *transfer buffer* gebruiken om de staat van de *controls* aan de hand van de loop van de applicatie te wijzigen. Elke *control* heeft een apart veld, dat ook gebruikt kan worden als de dialog niet geopend is.

Voor het gebruik van dit *transfer* mechanisme zijn de volgende 3 stappen noodzakelijk:

- Definiëren van een *transfer buffer*
- Definiëren van een corresponderende dialogbox of window
- Overdracht (*transfer*) van de data

Het *transfer* mechanisme wordt op diverse plaatsen binnen RAID toegepast. Zodra het programma wordt opgestart worden alle aanwezige dialogboxen gevuld met default instellingen. Dit zijn zowel de *default inputs* binnen *editcontrols* als de *default* selecties van de aanwezige checkboxen en/of radiobuttons. Deze (*default*) informatie wordt na het opstarten direct in de transfer records gezet van de betreffende dialogboxen. Wanneer door middel van een menu-optie een dialogbox geselecteerd wordt, krijgt de gebruiker dus de *default* dialogbox instellingen te zien en kan deze desgewenst ook wijzigen. Met het wijzigen verandert ook de inhoud van het transferrecord behorende bij een dialogbox.

Natuurlijk moet de bovenstaande dialogbox-input ook worden verwerkt. Dit wordt door diverse routines gedaan, die de velden van het betreffende transfer record uitlezen. Ook wordt het transfer record toegepast om dialog input naar disk te schrijven of van disk op te halen. Zo kan de gebuiker een gewenst scenario bekijken of wijzigen.

4.1.7 Het gebruik van units

Het gebruik van units vormt de basis van het modulair programmeren in Borland Pascal for Windows. Units worden gebruikt om libraries te creëren met objecten die in verschillende programma's (her)gebruikt kunnen worden. Een kenmerk van units is, dat de code (implementation gedeelte van een unit) afgeschermd kan worden, zodat de programmeur alleen toegang tot de code krijgt door middel van procedures en functions. Deze procedures en functions bevinden zich in het zogenaamde *interface* gedeelte van de unit.

Het unit principe wordt bij RAID toegepast om de overzichtelijkheid van het programma te verhogen. Dit wordt bereikt door de diverse objecten thematisch binnen units te groeperen ("plot-unit", "reken-unit"). Hierdoor wordt het makkelijker om objecten snel te vinden en te wijzigen.

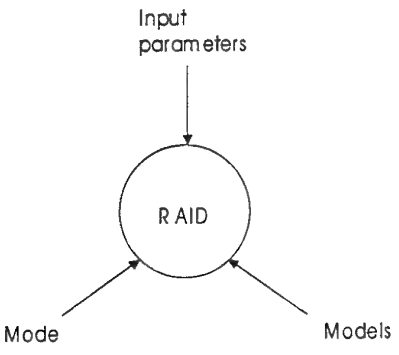
Voorbeelden van units binnen RAID zijn GenLib, een *general library* waarin algemene procedures en functies zijn ondergebracht (vergelijkbaar met de unit WinProcs van Borland Pascal for Windows), en twee units voor globale variabelen en constanten (Myvars en Const). Omdat deze laatste twee units globale data bevatten, mogen deze variabelen en constanten binnen alle andere units geïmporteerd worden.

4.2 Structurele opbouw van het programma

4.2.1 Blokdiagram

In zijn eenvoudigste vorm kan RAID worden voorgesteld als een black box die aan de hand van een drietal categorieën input een berekening uitvoert (figuur 4.7):

- *Input* parameters
- Geselecteerde meteorologische modellen
- De geselecteerde mode



Figuur 4.7: Programma-input

De input wordt door de gebruiker met behulp van dialogboxen ingevoerd. Zodra de invoer gereed is, kunnen (na verificatie van de input) de benodigde modelbereke-

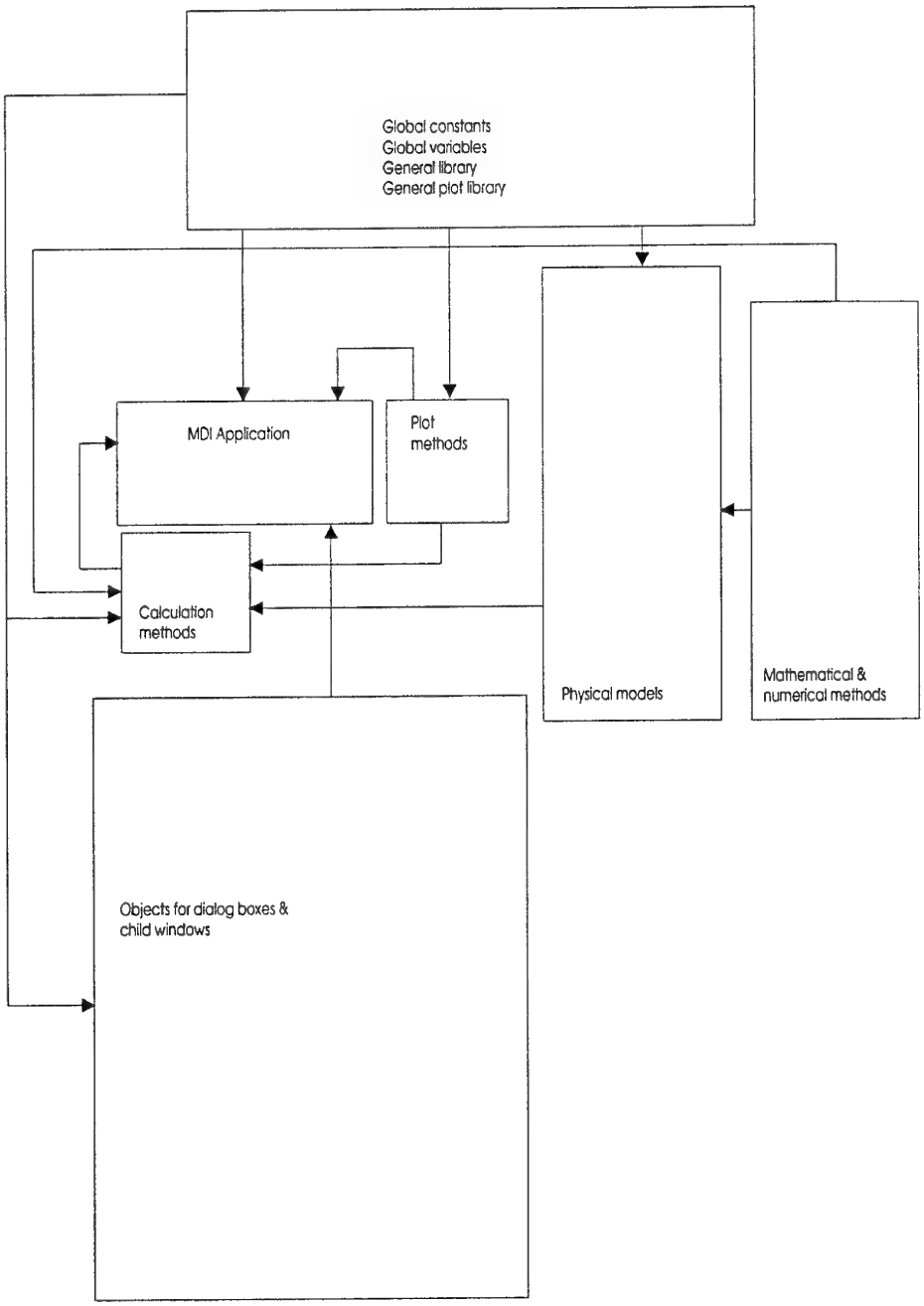
ningen worden uitgevoerd. Afhankelijk van de geselecteerde mode worden de resultaten in een bepaald type plot gepresenteerd.

In figuur 4.8 is in een blokdiagram de samenhang tussen de diverse programma-onderdelen weergegeven. Dit blokdiagram is onder te verdelen in totaal 7 blokken die elk één of meer units vertegenwoordigen en is als volgt ingedeeld:

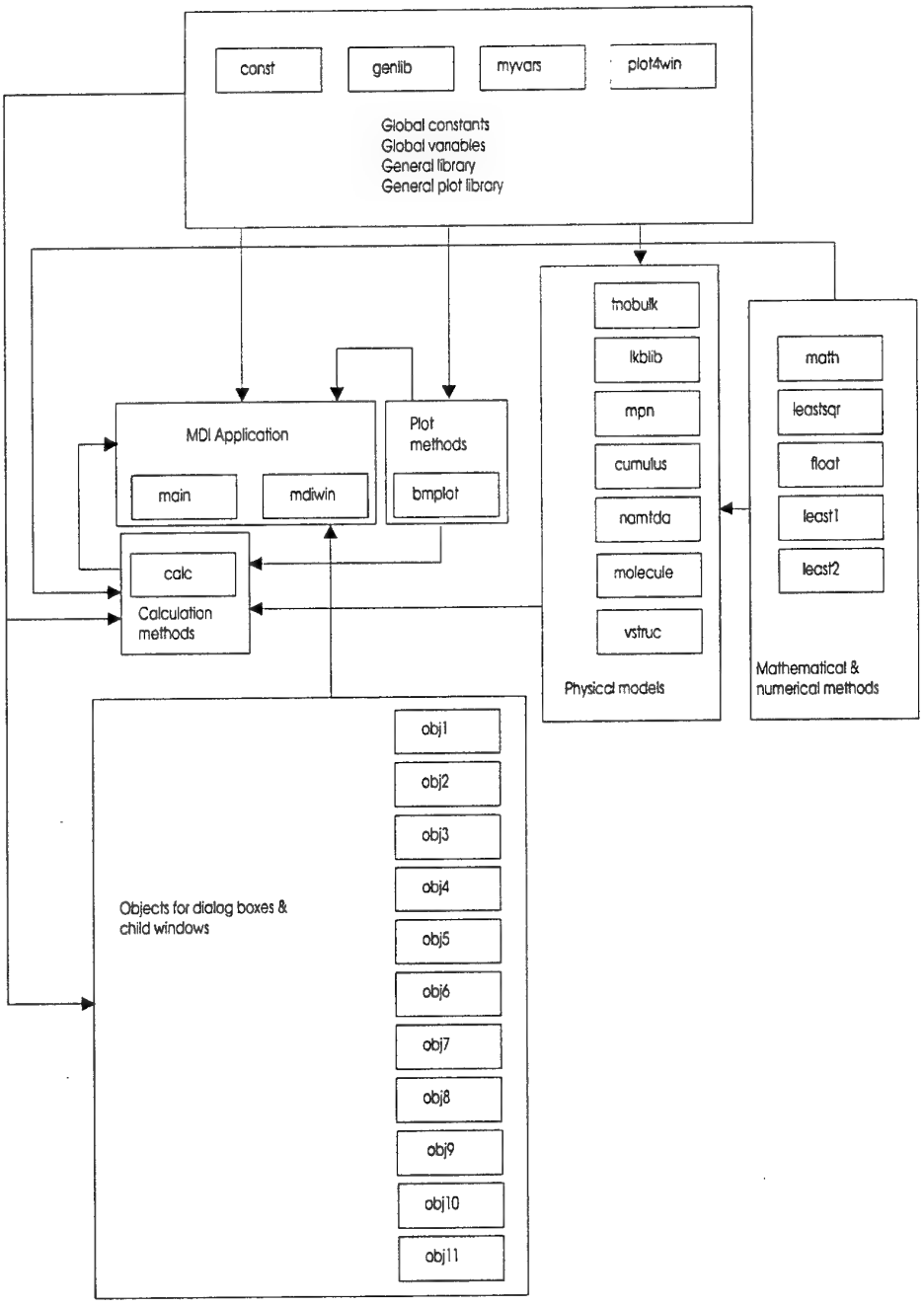
- Het bovenste blok (*Global constants, Global variables, General library* en *General plot library*) bevat algemene libraries voor code en data.
- De input wordt verzorgd met behulp van dialogboxen, waarvan de objecten zijn ondergebracht in het onderste blok (*Objects for dialog & child windows*).
- Nadat de input gegeven is, selecteert het *Calculation methods* blok de benodigde modellen en bepaalt welke *Plot methods* aangeroepen moeten worden. De modelberekening wordt gedaan door units uit het blok *Physical models*. Deze units importeren wiskundige en nummerieke methoden uit het *Mathematical & numerical methods* blok.

Figuur 4.9 geeft aan welke units te vinden zijn binnen de blokken van figuur 4.9. De naamgeving van deze units komt overeen met de naamgeving van de units uit de *source*-code. Dit figuur is vrijwel hetzelfde als figuur 4.8 met dien verstande dat hier meer details met betrekking tot de gebruikte units zijn weergegeven

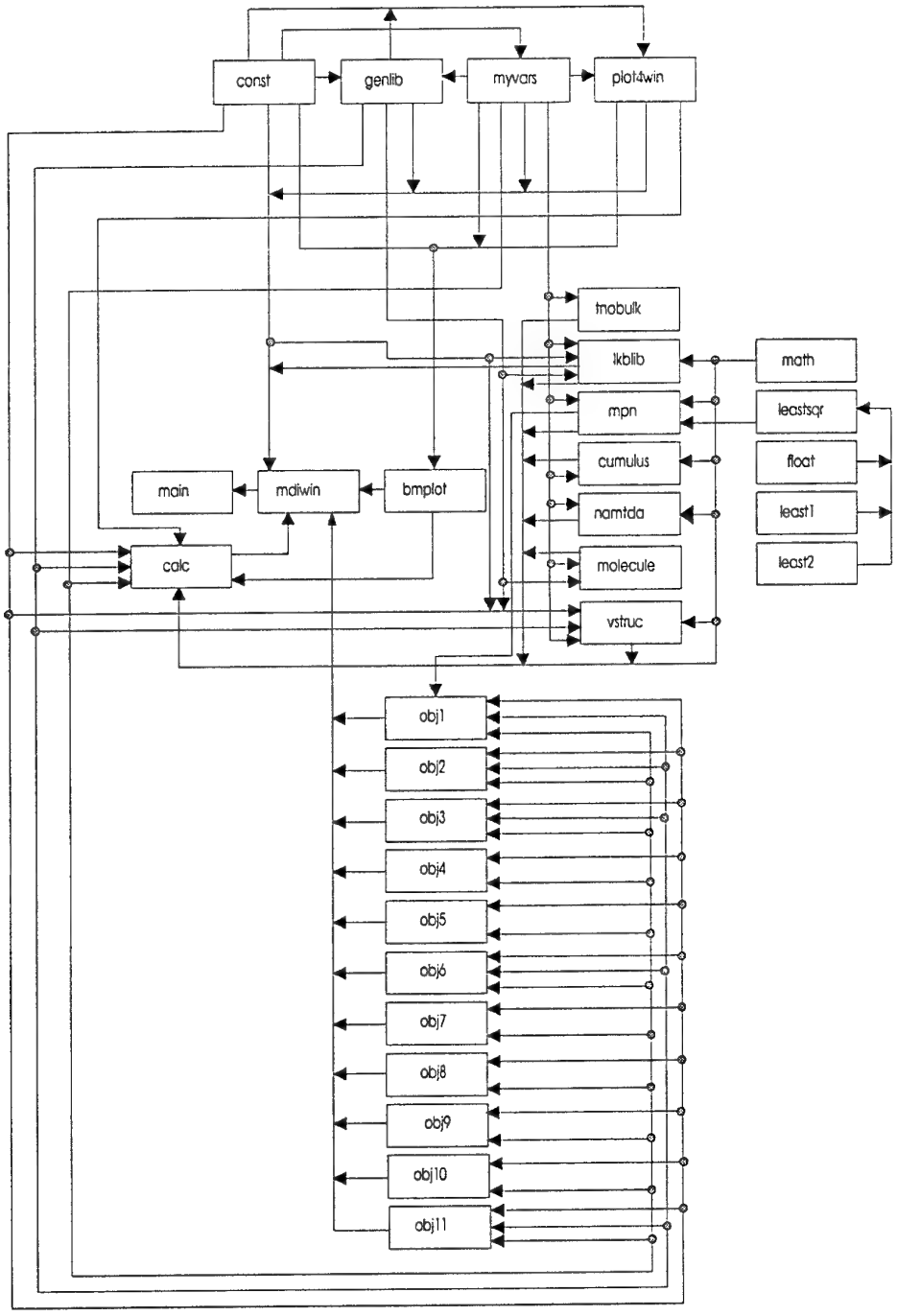
In Fig.4.10 is tenslotte de import en export van de aanwezige units te zien. Een pijl die een unit in gaat vertegenwoordigt de import van code of data. Een uitgaande pijl vertegenwoordigt de export van code of data.



Figuur 4.8: Blokdiagram



Figuur 4.9: Units

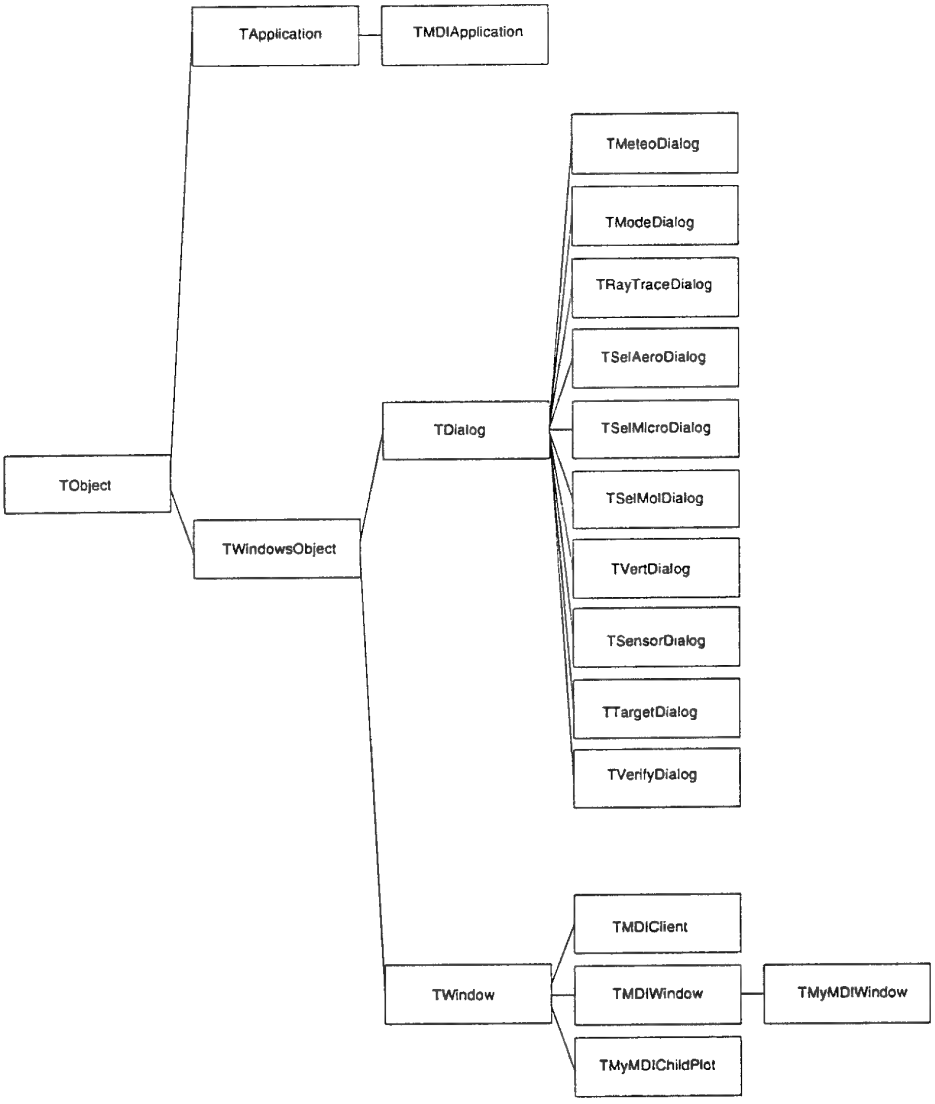


Figuur 4.10: Import / export

4.2.2 Hiërarchische indeling van de gebruikte objecten.

Figuur 4.11 toont de objecten die gebruikt worden in het programma RAID. Links bevindt zich het meest abstracte object TObject, dat de basis vormt van alle applicaties die voor Windows geschreven worden. Van links naar rechts worden de

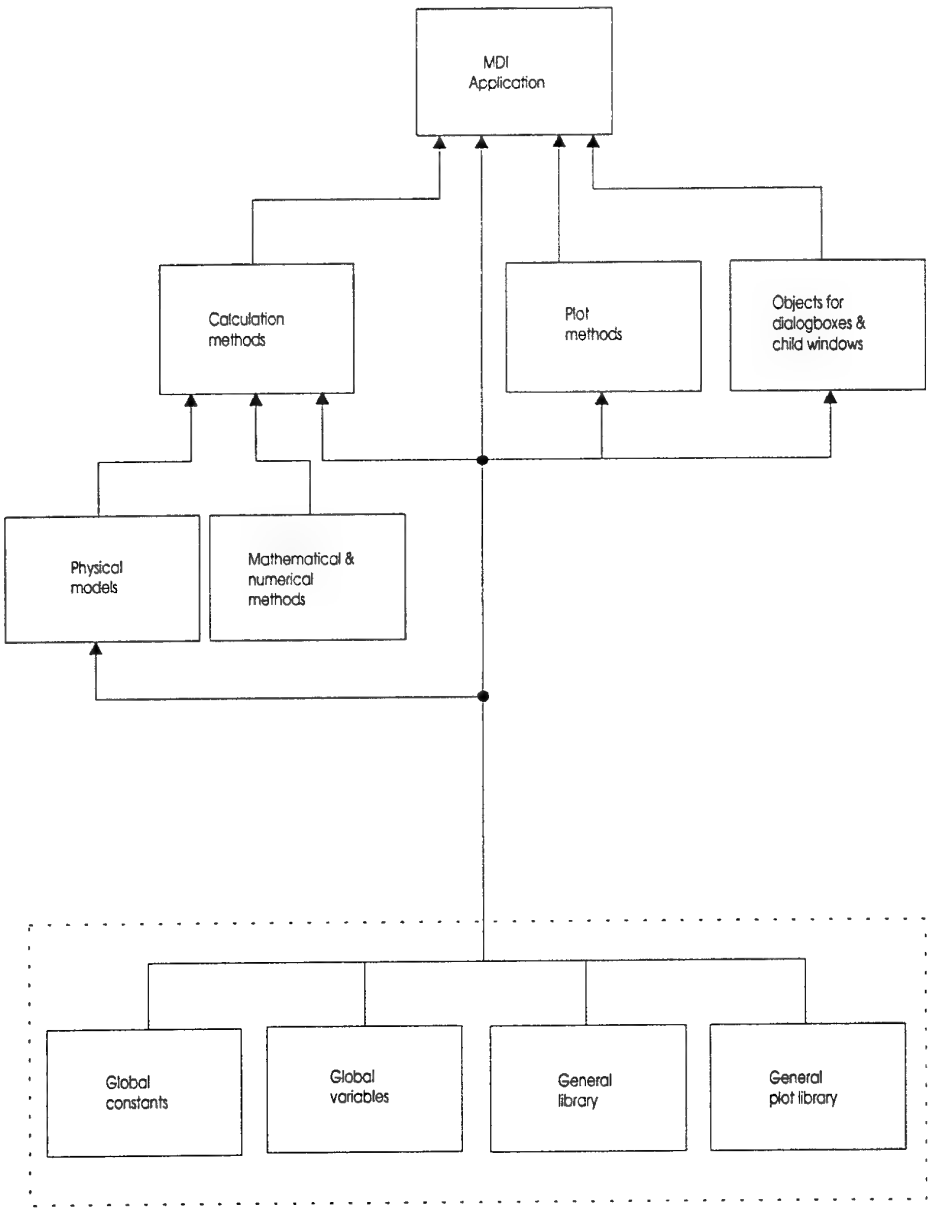
objecten steeds gedetailleerder (bijvoorbeeld TObject → TWindow → TMDIWindow). Uiterst rechts vinden we de concrete objecten voor het gebruiken van bijvoorbeeld dialogboxen.



Figuur 4.11: Gebruikte objecten

4.2.3 Hiërarchische indeling van de gebruikte units

De hiërarchische samenhang tussen de gebruikte units is in figuur 4.12 te zien (units van Borland Pascal for Windows worden hier buiten beschouwing gelaten). Elk blok stelt één of meer units voor. Units binnen één blok verrichten taken die passen bij de naamgeving van dat blok. Zo bevat het blok *Physical models* units die betrekking hebben op de modellen. De naamgeving van deze blokken komt overeen met de naamgeving die is gebruikt in het blokschema van figuur 4.8 en figuur 4.9.



Figuur 4.12: Hiërarchische opbouw

De hiërarchische samenhang in figuur 4.12 dient als volgt geïnterpreteerd te worden: units die zich in een hogere laag van de hiërarchie bevinden, kunnen units importeren die zich in dezelfde laag of lager in de hiërarchie bevinden.Units die zich in de onderste laag van de hiërarchie bevinden kunnen geen units meer importeren.

Het gestippelde kader in de onderste laag van de hiërarchie geeft aan dat het gaat om units die globaal gebruikt worden binnen het programma.Voor een beschrijving van deze units wordt verwezen naar paragraaf 4.3.1.

4.2.4 Opzet van de user-interface (GUI)

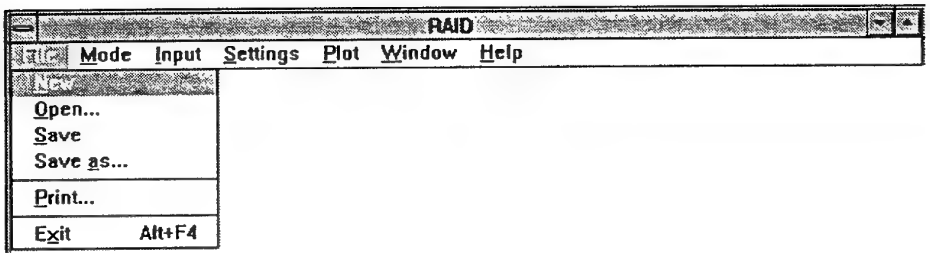
Het ontwerpen van een *Graphical User Interface*, kortweg GUI, is een vak apart. Omdat steeds meer programmatuur verschijnt die gebruik maakt van Windows is het wenselijk over een standaard 'look' te beschikken. Daarmee vindt de gebruiker telkens op dezelfde plaats bepaalde basisfuncties (bijvoorbeeld ophalen van bestanden, opslaan van bestanden en opvragen van hulp), zodat hij deze niet telkens opnieuw moet zoeken.

De geaccepteerde standaard is opgezet door IBM en wordt de *Common User Access* (CUA) genoemd. Deze standaard geeft bijvoorbeeld aan dat van links naar rechts op de menubar de opties File, Edit, Window en Help worden geplaats met eventuele extra opties tussen Edit en Window. De CUA geeft ook een volgorde voor de inhoud van bepaalde menu's zoals bijvoorbeeld File New, File Open, File Save, File Save As etcetera. Ook de betekenis van belangrijke toetsen (F1 voor help) is vastgelegd. Auteurs van een programma vermelden hun naam altijd onder de keuze About van de optie Help.

Met de resource workshop van Borland worden handige CUA *templates* aangeleverd. Het is belangrijk om deze templates te gebruiken, omdat de programmatuur zo volgens CUA standaard geleverd wordt. Het blijkt dat een programma dat op een andere manier basisfuncties presenteert irritatie oproept bij de eindgebruiker en dat deze het programma daardoor niet zal kiezen.

Het programma RAID voldoet volledig aan de CUA standaard van IBM. Als illustratie hiervan wordt in figuur 4.13 de (hoofd)menu bar getoond. Zoals vereist staat File links en Help rechts. Edit functies zijn in RAID niet geïmplementeerd, zodat deze optie op de menu bar ontbreekt.

Figuur 4.13: Hoofdmenu opties

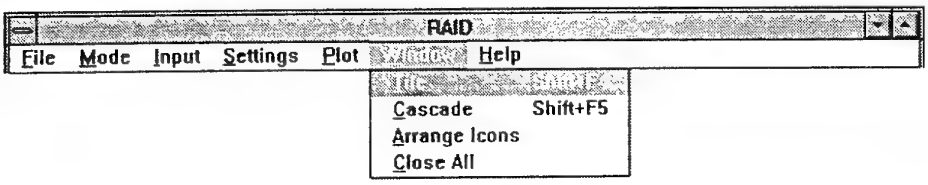


Figuur 4.14: Menu optie File

In figuur 4.14 zijn nog een tweetal andere kenmerken van de CUA-standaard te zien. Het gebruik van drie puntjes achter een keuze houdt in dat er een dialogbox volgt, waarin de gebruiker informatie dient te verstrekken of kan veranderen. De lijnen geven een scheiding aan tussen verschillende soorten keuzes. Hier is dat op

(vaste) volgorde het bewerken van een bestand, het afdrukken van een bestand en het verlaten van het programma. De keuze Exit wordt altijd onderaan het keuzemenu geplaatst, zodat de kans dat de gebruiker dit per ongeluk kiest klein is.

Een belangrijk onderdeel van de *user-interface* is het presenteren van de windows die informatie bevatten. Omdat iedere gebruiker informatie op zijn eigen manier gepresenteerd wil zien, zijn een aantal functies bedacht die hiervoor kunnen zorgen (zie figuur 4.15).



Figuur 4.15: Window opties

De keuzes zijn (in volgorde van CUA standaard):

- *Tile*: verdeel openstaande windows in evenredig grote stukjes.
- *Cascade*: plaats de openstaande vensters achter elkaar.
- *Arrange Icons*: bij gesloten MDI vensters, de icons netjes achter elkaar plaatsen.
- *Close All*: alle MDI windows sluiten.

4.2.5 Conventies naamgeving procedures / functies / vars etc.

In de vorige paragraaf is het belang van een standaard voor het opzetten van de user-interface besproken. Vreemd genoeg bestaat een dergelijke standaard (nog) niet voor het opzetten van de source code.

Voor kleinere programma's (tot zo'n 500 regels) is dat wellicht niet echt noodzakelijk, omdat meestal geen echte problemen te verwachten zijn met betrekking tot onleesbaarheid. Voor grotere programma's (meer dan 1000 regels) is een gestructureerde aanpak absoluut noodzakelijk.

Wegens het ontbreken van een algemene standaard zijn voor RAID eigen richtlijnen ontworpen, die hieronder worden besproken:

Algemeen:
Standaard Pascal uitdrukkingen (begin, end, for, while etc.) worden met kleine letters geschreven. Bij variabelen, functies en procedure headers is dit niet noodzakelijk en worden hoofdletters gebruikt om de leesbaarheid te verhogen (bijvoorbeeld 'TestProcedure').

Procedures, functies en methods worden op alfabetische volgorde opgenomen in de implementatie en interface secties van units. Waar dit niet mogelijk is wordt dit van duidelijk commentaar voorzien.

In de *uses* declaratie komt eerst een lijst van includes / units m.b.t. Borland Pascal for Windows. Daarna volgen de eigen units / includes op alfabetische volgorde.

Binnen een object worden de constructor en destructor als eerste gedeclareerd. Daarna volgen de betreffende *methods*.

Ieder *level* van programmatuur (if...then) wordt met inspringen verduidelijkt.

Commentaar:
Er zijn vijf soorten commentaar, die hieronder besproken worden. Aansluitend worden voorbeelden gegeven:

Programma header commentaar: geeft een duidelijke omschrijving over de versie, de datum, wijzigingen t.o.v de vorige versie.

Unit *header* commentaar: geeft informatie over de unit en verteld waarvoor de unit dient.

Scheidings commentaar tussen procedures / functies: zrgt voor een duidelijke weergave van procedures / functies en verhoogt de leesbaarheid van de *source* code.

Commentaar dat object methods onderscheidt van procedures / functies: maakt een duidelijke scheiding tussen methods / procedures en functies

Overig commentaar, binnen een procedure / functie / object: ter verduidelijking van de werking van de betreffende code.

1. Voorbeeld programma *header*:

```
{*************************************************************}
{                                                             }
{  Program : RAID                                             }
{       (RAnge forecast for Infra red Detectors)              }
{                                                             }
{  V100 M.Sannes / J.C. Vuijst                                }
{  Date : 01-12/1995                                          }
{                                                             }
{  Beschrijving :                                             }
{                                                             }
{  Toevoegingen t.o.v. vorige versie :                        }
{                                                             }
{*************************************************************}
```

2. Voorbeeld unit *header*:

```
{*************************************************************************}
{ UNIT Constant                                                           }
{                                                                         }
{ Hoofdprogramma : RAID.PAS                                               }
{                                                                         }
{ Unit met alle globale constanten                                        }
{                                                                         }
{*************************************************************************}
```

3. Voorbeeld scheiding tussen procedures:

```
procedure proc1;
begin
end;
```

twee lege regels

```
{ ------------------------------------------------------------------------------------------ }
```

één lege regel

```
procedure proc2;
begin
end;
```

4. De *methods* behorende bij objecten worden bij elkaar in het *implementation*-gedeelte gezet. Boven deze *methods* wordt het commentaar als volgt neergezet:

```
{ ------------------------------------------------------------------------------------------ }
{ method(s) behorende bij TSelAeroDialog                                                     }
{ ------------------------------------------------------------------------------------------ }
```

Een andere manier om methods te herkennen is hun formaat waarin een punt moet voorkomen (v.b. TSelAeroDialog.Init). De algemene vorm is 'Object.MethodName.'

5. De plaats van commentaar binnen de rest van het programma is niet vastgesteld. voorbeeld:

```
begin
  statement1;
```

```
  { hier krijgt de variable een waarde }

  a := 1;

  statement 2;
  statement 3;
  enz..
end;
```

4.3 Technische beschrijving van het programma

4.3.1 Gebruikte units en objecten

De gebruikte units en objecten worden beschreven volgens de hiërarchische indeling die is besproken in paragraaf 4.2.3.

hoofdprogramma MAIN.PAS (Blok: MDI Application)—Dit programma staat samen met unit MDIWIN in de bovenste laag in de hiërarchie. Alle Windows applicaties ontlenen een object type van TApplication om allereerst een main window op te bouwen van een door de gebruiker gedefiniëerd object type. Deze *instance* van Tapplication wordt binnen het hoofdprogramma gemaakt. Hier wordt tevens een resource bestand ingelezen dat gemaakt is met de resource editor van Borland Pascal for Windows. Met deze *editor* is het mogelijk om op een eenvoudige manier dialogboxen te maken en hierin *controls* te plaatsen zoals bijvoorbeeld (radio)buttons en editcontrols.

unit MDIWIN.PAS (Blok: MDI Application)—Deze (Object)unit staat boven in de hiërarchie omdat deze het object 'TMyMDIWindow' bevat. TMyMDIWindow reageert op *messages* die voortkomen uit de veschillende menu-opties.
De *message* id's (cm_- constanten) waarop dit object reageert, komen overeen met de id's die in de resource *editor* aan de menu-opties is meegegeven.

unit CALC.PAS (Blok: Calculation methods)—Binnen deze unit worden de benodigde modellen aangeroepen. De resultaten van de modelberekeningen worden vervolgens verwerkt in een plot (coördinaten van lijnen en symbolen). De berekende plot-coördinaten worden met behulp van procedures en functions uit unit PLOT4WIN.PAS in een bitmap geschreven, die vervolgens op het beeldscherm wordt gezet.

unit BMPLOT.PAS (Blok: Plot methods)—Deze unit bevat procedures die de plotomgeving initialiseren, zoals het neerzetten van tekst naast de plot en het aanroepen van plot-routines uit unit PLOT4WIN.PAS. De manier waarop een plot wordt geïnitialiseerd, hangt af van het soort plot dat moet worden getekend.

unit OBJ1.PAS (Blok: Objects for dialogboxes & child windows)—Deze object-unit onderscheidt zich van de andere object-units doordat deze geen object bevat voor een dialogbox, maar voor een child window. In dit child window wordt de plot met bijbehorende tekst gezet. Door verschillende *instances* van dit object te maken, kunnen meerdere plot windows tegelijk op het scherm gezet worden.

unit OBJ2.PAS (Blok: Objects for dialogboxes & child windows)—Deze object-unit bevat een object voor de 'Meteo Input' dialogbox. Dit object bevat *methods* voor afhandeling van checkbox messages. Naast de cancel en canclose *methods*, heeft dit object methods die controleren of de invoer (gegeven door middel van editcontrols) correct is. Binnen deze unit wordt het transfer buffer mechanisme toegepast, zoals is beschreven in paragraaf 4.1.6.

unit OBJ3.PAS (Blok: Objects for dialogboxes & child windows)—Deze object-unit bevat een object voor de 'Sensor Input' dialogbox die naast ok en cancel buttons nog radio buttons en edit controls bevat. Naast de cancel en canclose *methods*, heeft dit object *methods* die controleren of de invoer (gegeven door middel van editcontrols) correct is. Binnen deze unit wordt het transfer buffer mechanisme toegepast, zoals beschreven is in paragraaf 4.1.6.

unit OBJ4.PAS (Blok: Objects for dialogboxes & child windows)—Deze object-unit bevat een object voor de 'Mode' dialogbox die naast ok en cancel buttons nog radio buttons bevat. Dit object bevat geen *object fields* en geen *methods*.

unit OBJ5.PAS (Blok: Objects for dialogboxes & child windows)—Deze object-unit bevat een object voor de 'Select Aerosol Model' dialogbox die naast ok en cancel buttons nog *radio* buttons bevat. Dit object bevat naast de cancel en can-close *methods*, alleen *methods* die het geselecteerde aerosol model in het drop out menu onder de menubalk plaatsen.

unit OBJ6.PAS (Blok: Objects for dialogboxes & child windows)—Deze object-unit bevat een object voor de 'Target Input' dialogbox die naast ok en cancel *buttons* alleen edit controls bevat. Dit object bevat naast de cancel en canclose *methods*, alleen *methods* die kijken of de invoer (die gegeven is door middel van editcontrols) correct is. Binnen deze unit wordt het transfer buffer mechanisme toegepast, zoals beschreven is in paragraaf 4.1.6.

unit OBJ7.PAS (Blok: Objects for dialogboxes & child windows)—Deze object-unit bevat een object voor de 'Verify' dialogbox. Deze dialogbox geeft alleen informatie die aangeeft welke modellen te gebruiken zijn bij een bepaalde input van het input meteo dialogbox. Dit object bevat geen object-fields en ook geen *methods*.

unit OBJ8.PAS (Blok: Objects for dialogboxes & child windows)—Deze object-unit bevat een object voor de 'Select Molecular model' dialogbox die naast ok en

cancel buttons alleen radio buttons bevat. Dit object bevat naast de cancel en canclose *methods*, alleen methods die het geselecteerde molecular model in het drop-out menu onder de menubalk plaatsen.

unit OBJ9.PAS (Blok: Objects for dialogboxes & child windows)—Deze object-unit bevat een object voor de 'Select Vertical Structure' dialogbox die naast ok en cancel buttons alleen radio buttons bevat. Dit object bevat naast de cancel en canclose *methods*, alleen *methods* die het geselecteerde vertical structure in het drop out menu onder de menubalk plaatst.

unit OBJ10.PAS (Blok: Objects for dialogboxes & child windows)—Deze object-unit bevat een object voor de 'Select Micrometer' dialogbox die naast ok en cancel buttons alleen radio buttons bevat. Dit object bevat naast de cancel en canclose *methods*, alleen *methods* die het geselecteerde vertical structure in het drop out menu onder de menubalk plaatst.

unit OBJ11.PAS (Blok: Objects for dialogboxes & child windows)—Deze laatste object-unit bevat een object voor de 'Ray Tracing Input' dialogbox die naast ok en cancel buttons alleen edit controls bevat. Dit object bevat naast de cancel en canclose *methods*, alleen *methods* die kijken of de invoer (die gegeven is door middel van editcontrols) correct is. Binnen deze unit wordt het transfer buffer mechanisme toegepast, zoals beschreven in paragraaf 4.1.6.

De verschillende atmosferische modellen die binnen RAID worden gebruikt, bevinden zich in de volgende units (Blok: Physical models):

- unit TNOBULK.PAS
- unit LKBLIB.PAS
- unit MPN.PAS
- unit CUMULUS.PAS
- unit NAMTDA.PAS
- unit MOLECULE.PAS
- unit VSTRUC.PAS

Voor een beschrijving van deze modellen wordt verwezen naar hoofdstuk 2.

unit MATH.PAS (Blok: Mathematical & numerical methods)—Deze unit bevat een library van wiskundige functies, die gebruikt worden bij berekeningen binnen de atmosfeer modules.

unit LEASTSQR.PAS (Blok: Mathematical & numerical methods)—Deze unit (Borland) bevat numerieke methoden die toegepast worden binnen atmosfeer module MPN.

unit CONST.PAS (Blok: Global constants)—Deze unit bevat alle globale constanten. Deze constanten zijn onder te verdelen in:

- Constanten met betrekking tot menu opties (cm_-constanten)

- Constanten met betrekking tot dialog boxen (id_ -constanten)
- Constanten met betrekking tot default dialog instellingen (c_ -constanten)
- Constanten met betrekking tot dialogbox-invoer bereik (_min, _max - constanten)
- Constanten met betrekking tot kleur definities
- String constanten voor het invullen van menu opties
- Constanten ten behoeve van atmosfeer modules
- Overige constanten

unit MYVARS.PAS (Blok: Global variables)—Deze unit bevat globale declaraties van types en variabelen. Deze types en variabelen zijn onder te verdelen in:

- Definities en declaraties met betrekking tot arrays
- Definities en declaraties met betrekking tot transfer buffers
- Definities en declaraties met betrekking tot (tijdelijke) buffers
- Definitie en declaratie van het Monin-Obhukov record (LKB-model)

unit GENLIB.PAS (Blok: General library)—Deze unit bevat een reeks algemene procedures en functies die vanuit elke unit aangeroepen mogen worden.

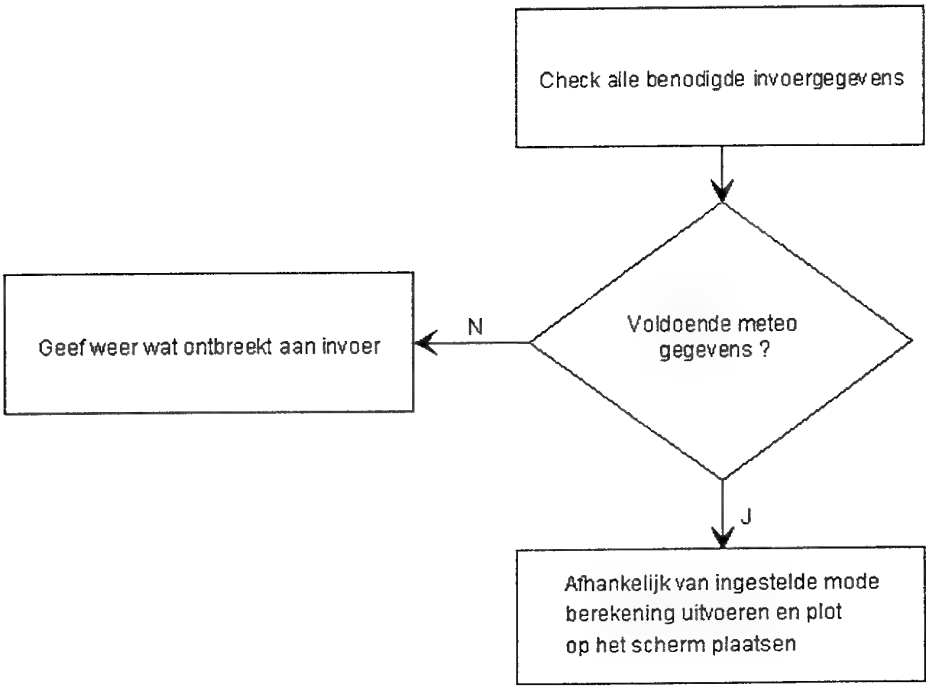
unit PLOT4WIN (Blok: General plot library)—Deze unit bevat routines voor het tekenen van grafieken, zoals bijvoorbeeld routines die assen en *grid* lijnen tekenen.

4.3.2 Belangrijke procedures en functies

De meest gecompliceerde procedures van RAID zijn IDPlot, CalcRayTracing en CalcDetProbMd, die respectievelijk plot, ray-tracing en detection probability en de 'supervisor' van de berekening / presentatie zijn. Deze procedures worden hieronder uitgebreider beschreven, waarbij ook de fysische achtergrond besproken wordt.

IDPlot

De procedure IDPlot verzorgt zowel de berekening en de weergave van de output van het programma. Telkens wanneer hoofdmenu optie Plot wordt aangeklikt, start de computer procedure IDPlot. Afhankelijk van de ingestelde mode en settings wordt een bepaald type berekening uitgevoerd en in een plot weergegeven. Figuur 10 geeft schematisch de werking van IDPlot weer:



Figuur 4.16: IDplot

Het bovenste blok in figuur 4.16 symboliseert het controleren van alle benodigde invoergegevens (zijn er voldoende meteorologische gegevens zijn ingevoerd om de geselecteerde modellen te kunnen gebruiken voor het maken van een berekening?). In het programma correspondeert dit met een lijst met sequentiële vergelijkingen. Als één van deze vergelijkingen onjuist is, wordt door middel van een if statement een verwijzing gemaakt naar de module die weergeeft welke input ontbreekt.

Het linkerblok in figuur 4.16 toont deze module. Bij aanroep wordt een grote matrix op het scherm geplaatst van meteorologische inputvelden en de modellen die gebruik maken van deze velden. Zo is te zien wat het gevolg is van ontbrekende input.

Indien alle benodigde input voorhanden is, wordt de berekening gestart en vervolgens de resultaten weergegeven. In figuur 4.16 is dit onderdeel gesymboliseerd in het onderste blok. Het onderste blok is te vervangen door een aantal (gedetailleerdere) blokken:

Figuur 4.17: Detail IDPlot

Zoals te zien is in figuur 4.17 worden vier functies achtereenvolgens uitgevoerd. Allereerst wordt een bitmap in het geheugen aangemaakt om deeluitkomsten van de berekening direct op een (virtueel) scherm te kunnen plaatsen. Voor deze methode is gekozen omdat dit een van de gunstige oplossingen is gebleken bij de combinatie van veel rekenwerk en het verplaatsen van windows. Zie hiervoor het gedeelte over bitmaps in paragraaf 4.1.5.

De tweede functie (tweede blok van boven in figuur 4.17) verzorgt het feitelijke rekenwerk. Resultaten van deelberekeningen worden opgeslagen in de bitmap in het geheugen. Op het rekenwerk wordt hier niet ingegaan, dit staat uitgebreid beschreven in hoofdstuk 2.

In het derde blok wordt bepaald in welk window de bitmap geplaatst wordt. Iedere plot die gemaakt wordt met het programma wordt voorzien van een nummer en een titel die aangeeft om wat voor soort plot het gaat. Bij de nummering wordt het laagst beschikbare nummer gekozen. Dus, indien de gebruiker drie plots maakt (genummerd 1 t/m 3), vervolgens plot (window) 2 sluit, en tenslotte nog een vierde plot maakt, krijgt deze plot het vrijgekomen nummer 2.

In het laatste blok wordt procedure MakeWindow van Borland Pascal aangeroepen waarin de bitmap uit het geheugen, die *compatible* is met het grafische scherm, in het window geplaatst.

Ray tracing

De lichtsnelheid hang af van het medium. In een medium, anders dan vacuüm, is de lichtsnelheid steeds kleiner dan 3,0 10^8 m/s. De verhouding van de lichtsnelheid in een medium en in vacuüm wordt de brekingsindex genoemd en aangeduid met de letter n. De brekingsindex is dus dimensieloos en een getal groter of gelijk aan 1. Dikwijls is de brekingsindex nog een functie van de golflengte (kleur van het licht).

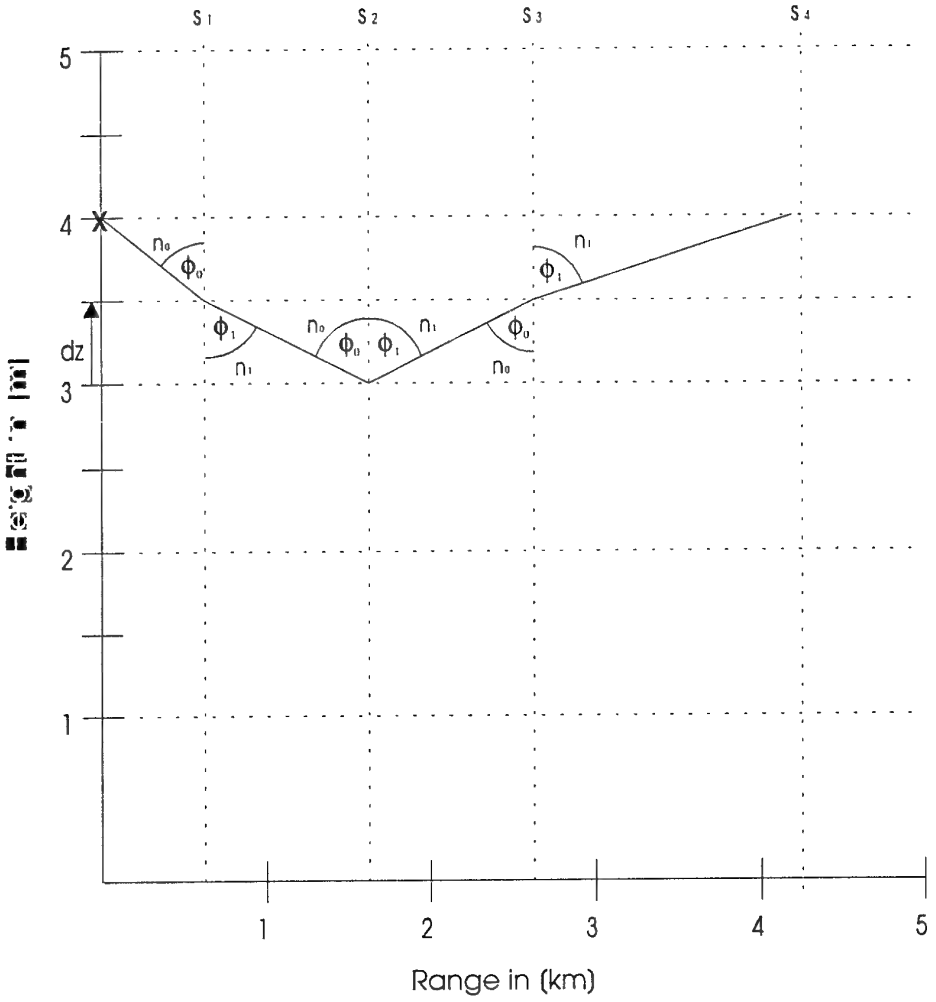
Op de overgang tussen twee media kan er ten gevolge van het verschil in de brekingsindices breking van licht optreden (zoals bijvoorbeeld bij een lepeltje in een kopje thee). Van dit verschijnsel wordt handig gebruik gemaakt in optische hulpmiddelen zoals lenzen, prisma's maar ook optische glasvezels. De brekingsindex voor water is ongeveer 1,3 en voor glas ongeveer 1,5. Er zijn ook materialen met een zeer hoge brekingsindex (3-5).

Lucht heeft een brekingsindex die slechts weinig, in de orde van grootte van 10^{-6}, verschilt van 1. Meestal wordt daarom de refractie opgegeven, aangeduid met N, die afgeleid is van de brekingsindex volgens $N= (n-1)x10^6$. Temperatuur- en/of vochtigheidsverschillen tussen opeenvolgende luchtlagen zorgen voor kleine variaties in de refractie. Deze kleine verschillen leiden tot aardige effecten zoals een elliptische (ondergaande) zon, een hoog uitgerekte boot, waarneming tot voorbij de horizon (of juist een beperking tot minder dan de geometrische horizon) of luchtspiegelingen.

Voor het berekenen van effecten van breking van licht bij de overgang tussen twee media, wordt gebruik gemaakt van de wet van Snellius. De wet van Snellius zegt dat het product van de brekingsindex en de sinus van de hoek van inval (met de normaal) constant is.

$n_0 \sin \Phi_0 = n_1 \sin \Phi_1$

De normaal is een denkbeeldige lijn op het oppervlak tussen de media. In het algemeen zijn de brekingsindices n_0 en n_1 bekend en kiezen we de hoek waaronder het licht invalt, Φ_0 zelf. De hoek waaronder het licht zich in het aanliggende medium voortplant laat zich nu met de formule berekenen. In figuur 4.18 is een voorbeeld te zien waarbij één lichtstraal wordt gebroken door de horizontaal gelaagde atmosfeer. De dikte van de atmosferische laagjes komt overeen met de stapgrootte in de z-richting (dz).



Figuur 4.18: Ray tracing

Ray tracing is belangrijk wanneer bepaald moet worden of een doel dat zich op een bepaalde plaats bevindt door een sensor kan worden waargenomen. Er moet dan worden bekeken of het licht dat door het doel wordt uitgezonden via het een of andere pad (met breking!) bij de sensor kan komen. Bij het berekenen van ray traces wordt vaak voor het rekengemak de richting van de lichtstraal omgekeerd. De lichtstralen gaan dan niet naar de sensor toe, maar worden door de sensor gegenereerd. Deze 'truc' veroorzaakt voor het resultaat geen verschil.

De implementatie van ray tracing is te vinden in procedure CalcRayTracing (unit: Calc). Bij het algorithme voor ray tracing wordt gebruik gemaakt van de volgende parameters:

- temperatuur- en vochtigheidsverschillen
- Sensor hoogte
- Gemiddelde hoek waaronder de lichtstralen de sensor binnenkomen
- Spreiding van inkomende lichtstralen

- Aantal stralen dat getekend moet worden

De gebruiker kiest een sensorhoogte, een starthoek en spreiding, alsmede het aantal lichtstralen. Het programma berekent vervolgens de benodigde temperatuur- en vochtigheidsverschillen en kiest een stapgrootte in de z-richting gekozen (integratie stap). Voor elke dz wordt de plaats (Range-richting) berekend waar de lichtstraal het interval verlaat (figuur 4.18 geeft dit aan met de normaal: S_1. Met de formule van Snellius wordt de hoek berekend waaronder de lichtstraal het volgende laagje binnenkomt (Φ_1). Door dit te herhalen kunnen de berekende *range* en *height* posities worden gemarkeerd in de plot. Er wordt gestopt met het tekenen van één lichtstraal als de maximale *range* en/of *height* bereikt zijn. Om de volgende lichtstralen te tekenen wordt bij de beginhoek van de huidige lichtstraal steeds de stapgrootte van de hoek voor de lichtstralen opgeteld.

Detection Probability
De procedure Detection Probability (de kans om gedetecteerd te worden) geeft een inschatting van de kans dat de sensor het doel zal waarnemen. Deze kans is afhankelijk van een aantal effecten die in de atmosfeer spelen, waarop we nu nader ingaan.
Wanneer licht zich door de atmosfeer voortplant, treden er verliezen op ten gevolge van absorptie en verstrooiing door aerosolen en moleculen. De transmissie (of extinctie) is een maat voor de hoeveelheid licht die overblijft en wordt vooral bepaald door het aantal aerosolen (en moleculen) in de atmosfeer. Op zijn beurt is dit aantal afhankelijk van de meteorologische condities van de atmosfeer (windsnelheid, temperatuur en vochtigheid). Doordat deze condities variëren met de hoogte, dient ook de verticale structuur van de atmosfeer bekend te zijn. De kans dat een doel door de gebruiker kan worden waargenomen (of andersom!) wordt dus bepaald door de atmosferische transmissie. Ook andere effecten zoals refractie en turbulentie spelen een rol, maar die worden in deze procedure buiten beschouwing gelaten.

Helaas blijkt het berekenen van het verticale profiel van de aerosol concentratie veel rekentijd te vergen. Daarom zijn er twee methoden geïmplementeerd om de detectie kans te berekenen:

- 'Snelle' methode: bereken geen verticale structuur van de transmissie. Dit wil zeggen dat de transmissie alleen wordt uitgerekend op een hoogte van 10 meter.
- 'Nauwkeurige' methode: bereken wel een verticale structuur, bijvoorbeeld met het NPS model. Dit wil zeggen dat de transmissie voor elke integratie stap (van sensor naar doel) opnieuw wordt bepaald.

Hieronder worden beide methoden in een programmafragment weergegeven. Omdat methode 2 nauwkeuriger is, is de complexiteit van dit fragment groter. Beide fragmenten bevatten pseudo-code ter verduidelijking.

Methode 1:

```
procedure PlotDetProbWithoutNPS (var HMemDC : HDC);
begin
   T := 1;                                  { Totale transmissie 100% }
   RangePart := 1                           { Range stapgrootte in km }

   bereken αT                               { Totale extinctie }
   bepaal fout in  αT                       { fout in extinctie per stap }

   for j := 1 to RangeMax do                { Maximale Range b.v. 40 km }
   begin
      T := T * exp(-αT * RangePart);        { Transmissie per stap }
      { Teken pixel met kleur afhankelijk van  transm. op plaats (j, 10.0) } }
      PlotTrans (HMemDC, j, 10.0, Trans);
   end; { for }
end;
```

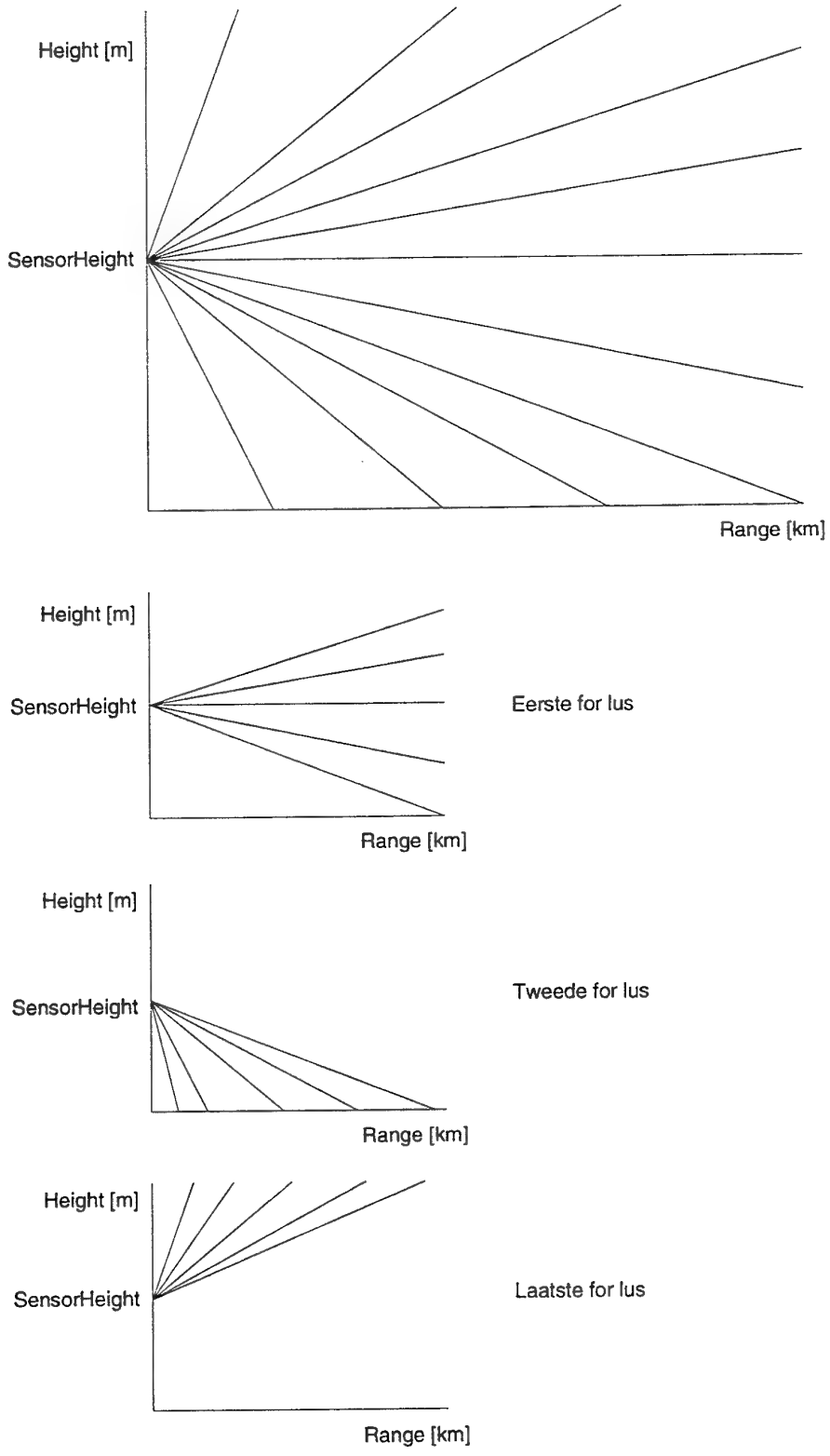
Methode 2 is aanzienlijk uitgebreider:

```
procedure PlotDetProbWithNPS(var HMemDC : HDC);
begin

   { bereken en teken alle transmissielijnen voor alle Target hoogtes }

   for HeightCount := 0 to Round(TargetHeightMax) do
   begin
      { teken 1 transmissielijn, behorende bij 1 target hoogte }
   end; { for }

   { bereken en teken alle transmissielijnen voor elke rangestap tussen
     sensorhoogte en z = 0 }

   for RangeCount := 0 to Round(RangeMax) do
   begin
      { teken 1 transmissielijn, behorende bij 1 sensor hoogte }
   end; { for }

   { bereken en teken alle transmissielijnen voor elke rangestap tussen
     sensorhoogte en z = TargetHeightmax }

   for RangeCount := 0 to Round(RangeMax) do
   begin
      { teken 1 transmissielijn, behorende bij 1 sensor hoogte }
```

```
    end; { for }

    bepaal gemiddelde fout in extinctie  { gem. foutwaarde over stappen }
end;
```

Het essentiële verschil tussen methodes 1 en 2 is dat bij methode 2 de extinctie voor elk benodigd hoogte-interval moet worden berekend, terwijl bij methode 1 slechts één extinctie (voor 10 meter hoogte) dient te worden berekend. Daarnaast moet in methode 2 rekening worden gehouden met de (hoogte)grenzen van de plot

Methode 2 is te ontleden in drie stukken, elk verantwoordelijk voor het berekenen van een onderdeel van de Detection Plot (zie figuur 4.19). De eerste for lus zorgt voor het tekenen van de transmissielijnen vanaf de sensor tot en met de maximale Range van de plot (40 km). In de tweede for lus worden de transmissielijn getekend die voordat de maximale range bereikt wordt reeds het oppervlak bereikt hebben. De laatste for lus tekent de transmissielijnen die de maximale plothoogte bereiken vòòr de maximale range bereikt wordt. Bij iedere nieuwe berekening wordt een nieuwe α_T berekend, en een nieuwe foutwaarde. De gemiddelde foutwaarde wordt aan het einde berekend.



Figuur 4.19: Detection probability

4.3.3 Verhelpen van errors binnen Borland Pascal for Windows

Tijdens de ontwikkeling van RAID zijn een aantal *bugs* ontdekt binnen de Borland omgeving. De meeste *bugs* staan in verband met het gebruik van arrays binnen dialog boxen. Een verklaring van dit soort *bugs* is tijdens de ontwikkeling van RAID niet gevonden. Binnen deze paragraaf zal besproken worden welk 'vreemd gedrag' zich kan voordoen en hoe de programmeur hiervoor een (nood)oplossing kan implementeren.

Vreemd gedrag bij het gebruik van array's—Binnen unit OBJ2.PAS bevindt zich de *method*: TMeteoDialog.MeteoInputCorrect die alle meteo invoer op juistheid controleert. Is alle invoer juist, dan wordt *true* terug gegeven. Binnen deze method is een array gedeclareerd (OutStr) van het type arr80char. Deze array is nodig om een melding in een message box te zetten wanneer de invoer onjuist is. Zodra dit array wordt vergroot naar bijvoorbeeld een array van het type arr100char, onstaat het probleem dat de ok button in de message box niet meer functioneert.

Dit probleem deed zich voor op een PC (80486 / 50Mhz processor met 8MB RAM). Als op deze PC het OutStr array (arr100char) wordt verkleind naar een array van het type arr90char, is dit probleem opgelost. Later werd ontdekt dat deze instelling nog problemen gaf op een andere PC (80386 / 25Mhz processor met 4MB RAM). Op deze PC werkte met dit array (arr90char) de cancel button van de betreffende dialogbox niet meer. Ter voorkoming van deze problemen moet de lengte van array OutStr tot 80 characters beperkt blijven. Met deze arraylengte zijn tot dusver nog geen problemen ontstaan.

Andere problemen—Binnen unit PLOT4WIN.PAS bevindt zich procedure Draw-Axis. Deze procedure bevat code om de assen van een grafiek op de juiste manier te plaatsen. In deze procedure moet het *font* (MyLogFont) ingesteld worden. Als dit niet wordt gedaan, kan het *font* veranderen, zodra windows worden vergroot of verkleind.

Bij het plaatsen van tekst naast de plot is ontdekt dat alleen true type *fonts* geroteerd en 'gescaled' kunnen worden. Met niet-true type fonts werken de betreffende commando's niet.

Als een OK push button van een dialog box ervoor moet zorgen dat de dialogbox verdwijnt, moet deze button in de resource workshop (van Borland) als *default* push button zijn opgegeven.

De Enter toets (staat voor het indrukken van de OK push button) resulteert bij alle input dialogboxen in het *closen* van de dialog box, behalve bij de input target dialogbox. Hier werkt het soms wel, soms niet. Toch is ook hier de OK push button als *default* in de resource workshop opgegeven. Met de muis OK aan klikken werkt echter wel altijd.

Het is niet altijd mogelijk om de input meteo dialogbox te *closen* door met de muis twee maal te klikken op de button in de linker bovenhoek. In bepaalde gevallen komt de message blijkbaar niet door.

4.3.4 Instellingen compiler van Borland Pascal for Windows

Bij RAID zijn de volgende instellingen noodzakelijk :

- Options / Compiler / Numeric Processing : 8087 - 80287 aan.
- Options / Compiler / Runtime Errors : alle checks aan.
- Options / Tools... / Turbo Debugger, arguments : -wd (Voor DEBUGING)
- Options / Memory sizes / alle stack sizes op : 16384
- Options / Environment / Editor / TabSize : 2

5. Aanbevelingen

5.1 Optimalisatie van de User Interface

Een aantal opties worden weliswaar in de menu's van RAID al genoemd, maar zijn nog niet geïmplementeerd. Het betreft hier:

- Menu opties print/printer setup: met de optie print moet een 'screen dump' van het client-area naar een printer gestuurd kunnen worden, zodat de diverse plots eenvoudig toegevoegd kunnen worden aan rapportages, verslagen en eventueel publicaties.
- Een lijst van recent gebruikte files zou kunnen worden toegevoegd onder menu-optie exit. Zo'n lijst bevindt zich al onder hoofd-menu optie 'Window' waarin een lijst van alle geopende windows te vinden is.
- De modes T vs Sensitivity, T vs Temperature Difference en T vs Height moeten nog geïmplementeerd worden.
- Alle help-opties moeten nog geïmplementeerd worden. Dit geldt voor zowel de help-opties van het hoofdmenu als voor de implementatie van help-buttons.

Bij het (test-)gebruik van RAID bleek dat de presentatie van het programma op een aantal punten verbeterd kan worden. Het betreft hier:

- Toevoeging van een toolbar(boven) of een statusbar(onder) zou meer duidelijkheid kunnen geven aan de gebruiker.
- De Verify Model Input dialogbox (presentatie van gevolgen ontbrekende meteorologische parameters) bevat momenteel een tabel waarin 5 modellen zijn ondergebracht. Bij toevoeging van modellen in een latere versie zou het handig zijn om van deze dialogbox een child-window te maken, met daarin horizontale en verticale scroll-bars. Met deze toevoeging kan een onbeperkt aantal modellen worden toegevoegd binnen het verify-overzicht. Ook kan met deze aanpassing het aantal parameters binnen dit overzicht worden vergroot.
- Tijdens het maken van de berekening verschijnt 'Calculating' in het midden van het window. Dit kan worden vervangen door een dialogbox waarin een progress-bar aangeeft hoever de berekening is gevorderd.
- De kleuren van de color-bar (Detection probability) zouden beter in elkaar moeten kunnen overlopen. Hiervoor moet een nieuw kleuren-palet worden ontworpen.
- Het zou wenselijk kunnen zijn dat de schaalverdeling van de gebruikte assen veranderd kan worden. Hiervoor moet dan een menu-optie worden toegevoegd, waarin de gebruiker bijvoorbeeld lin/log assen kan instellen.

De presentatie van de detection probability in false colors is gecompliceerd in de zin dat het niet in één oogopslag duidelijk is of een doel wel of niet gedetecteerd kan worden. Het is daarom wenselijk een vereenvoudigde presentatie te introduceren (een toepasselijke naam zou zijn: 'commander briefing') waarin duidelijk wel/niet kans op detectie van een bepaald doel wordt aangegeven.

5.2 Verbetering van de voorspelling

In deze versie van RAID wordt de detection probability uitgerekend met een rechte stralengang tussen sensor en (mogelijke) doelen. Deze mode dient derhalve geïntegreerd te worden met de mode Ray Tracing om een correct beeld te krijgen van de transmissie tussen sensor en doel. Deze aanpassing zal echter de rekentijd van het programma doen toenemen.

Bij het berekenen van de stralengang (Ray Tracing) wordt geen rekening gehouden met de kromming van de aarde. Hiervoor zou een analogie van de wet van Snellius gebruikt moeten worden, geldig voor spherisch gelaagde media. Indien echter de stapgrootte langs het lichtpad voldoende klein wordt genomen (maximaal 1 km), is het verschil tussen beide methoden te verwaarlozen.

De reikwijdtevoorspeller maakt voor de micrometeorologie gebruik van het LKB-model [Liu et al, 1979]. Recentelijk is een meetcampagne op MeetPost Noordwijk uitgevoerd in het kader van SIRIUS om data over de atmosferische point spread functie en refractieverschijnselen te verkrijgen [De Leeuw et al., 1995a]. Deze data is gebruikt om een nieuw micrometeorologisch model van Kunz [1995] te valideren, dat op een aantal punten een verbetering is van het LKB-model. Het model van Kunz zal dan ook in RAID worden geïmplementeerd.

Een aantal atmosferische effecten op het gebied van propagatie van (infrarode) straling zijn nog niet in RAID geïmplementeerd. Met name moet hier gedacht worden aan een module voor het berekenen van optische turbulentie (scintillatie en beam wander). Deze effecten kunnen berekend worden uit de zogenaamde brekingsindex structuurfunctie parameter $C_n^2(z)$, die beschikbaar is uit het model van Kunz [1995].

Tenslotte kunnen modules geïmplementeerd worden die invloeden op de performance van de sensor beschrijven. Hierbij kan gedacht worden aan een 'zon/maan-module', die afhankelijk van locatie, tijd en bewolking een lichtbron in de scene plaatst en een achtergrond module. Dit soort modules valt strikt genomen echter buiten het kader van het atmosferische deel van de reikwijdtevoorspeller, welke immers tot doel heeft de propagatie-eigenschappen van de atmosfeer te berekenen.

5.3 Optimalisatie van gebruikte numerieke methoden

De rekentijd voor versie 1.0 van RAID ligt in de orde van 30 seconden tot 2 minuten op een HP Vectra 486, CPU 66 MHz. Het merendeel van de rekentijd wordt besteed aan het model voor de verticale structuur en de ray-tracing.

Het model voor de verticale structuur van de aerosol-extinctie vereist een numerieke integratie tussen het oppervlak en de maximale hoogte in het domein. Deze

integraal wordt momenteel met een Simpson-algoritme berekend, waarbij het aantal integratie-intervallen in elke slag een factor twee verhoogd wordt tot de vereiste precisie is bereikt. Het aantal intervallen (startwaarde, increment per slag) is nog niet geoptimaliseerd. Daarnaast kan een andere numerieke integratiemethode geprobeerd worden.

De ray-tracing vergt veel rekentijd doordat het hele lichtpad in kleine stapjes moet worden doorlopen. Momenteel gebeurt dat met een robuuste stapgrootte, welke zo klein gekozen is dat een betrouwbare berekening wordt gegarandeerd. Het is mogelijk een variabele stapgrootte te introduceren als functie van de elevatiehoek van de lichtstraal. Daarnaast zou het gebruik van look-up-tables voor aerosolextinctie en meteorologische parameters op een verticaal (logaritmisch) grid overwogen kunnen worden. Tijdens het volgen van de lichtstraal kan dan volstaan worden met interpolatiewaarden uit een dergelijke tabel.

5.4 Uitbreiding van het pakket

Binnen het kader van opdracht A92KM613 is een op zichzelf staande atmosferische module voor een IR reikwijdtevoorspeller ontwikkeld. Voor een krachtige reikwijdtevoorspeller is het wenselijk om deze module te integreren met modules die betere beschrijvingen van sensoren, doelen en achtergronden geven.

Uiteindelijk kan een pakket ontstaan dat in een groter geheel functioneert:

- Met behulp van OLE (Object Linking and Embedding) functies kan een link gemaakt kunnen worden tussen de uitkomsten van de berekeningen en een database. Zo kan (bijvoorbeeld) een plot-icon worden 'gesleept' naar een database om de berekeningen te analyseren binnen een database of tekstverwerker.
- Door een koppeling tussen sensoren en de computer kan het actuele meteorologische scenario direct worden doorgegeven aan de reikwijdtevoorspeller
- Als functie van de geografische positie (GPS) kan de reikwijdtevoorspeller zelf de juiste modellen voor een berekening kiezen
- Een ontwikkeling tot expertsysteem maakt het mogelijk dat de reikwijdtevoorspeller aangeeft hoe de reikwijdte van een bepaald systeem vergroot kan worden (keuze andere golflengte).

6. Dankwoord

De auteurs zijn Gerard Kunz en Marcel Moerman erkentelijk voor hun bijdrage op het gebied van het refractiemodel en de implementatie daarvan in de reikwijdte-voorspeller.

De werkzaamheden van Michael Sannes en Jonathan Vuijst vonden plaats in het kader van een afstudeerproject van de Hogeschool Haarlem. De auteurs willen op deze plaats graag ook de prettige sfeer noemen waarin dit project is uitgevoerd.

7. Referenties

[1] Davidson, K.L., Frederickson, P.A. and De Leeuw, G. (1994). Surface layer turbulence and aerosol profiles during MAPTIP. The 2nd symposium of the SPP Panel of AGARD on 'Propagation assessment in coastal environments', 19-23 Sept 1994, Bremerhaven, Germany, CP-567.

[2] Davidson, K.L., DeCosmo, J., De Leeuw, G., Edson, J.B., Katsaros, K.B., Mengelkamp, H.T., Oost, W.A., Smith, S.D. en Taylor, P.K. (1995). Variation of wind, stress, heat and vapor fluxes in vicinity of fronts. In preparation.

[3] Deardorff, J.W. (1976). On the entrainment rate of a stratocumulus-topped mixed layer. Quarterly Journal of the Royal Meteorological Society, 102, 563-582.

[4] De Leeuw, G. (1986). Vertical profiles of giant particles close above the sea surface. Tellus 38B, 51-61.

[5] De Leeuw, G., Parashar, S., Park, P.M., Perry, S.J., Roney, P.L. en Smith, M.H. (1986). Aerosol Study in the North Atlantic 1983. NATO AC/243 (Panel IV/RSG.8) Report 1986-01.

[6] De Leeuw, G. (1989). Modeling of extinction and aerosol backscatter profiles in the marine mixed layer. Applied Optics, 28, 1356-1359.

[7] De Leeuw, G., van Eijk, A.M.J. en Noordhuis, G.R. (1993). Modeling aerosols and extinction in the marine atmospheric boundary layer. Atmospheric Propagation and Remote Sensing II. A. Kohnle and W.B. Miller (Eds.), Proc. SPIE 1968, 70-80.

[8] De Leeuw, G., van Eijk, A.M.J. en Jensen, D.R. (1994). MAPTIP experiment, Marine Aerosol Properties and Thermal Imager Performance: An overview. TNO rapport FEL-94-A140.

[9] De Leeuw, G., Schwering, P.B.W., Fritz P.J., Moerman, M.M., Kunz G.J., Cohen, L.H. en de Jong, A.N. (1995a). Atmospheric effects on detection of point targets at long ranges. TNO rapport, in voorbereiding.

[10] De Leeuw, G., van Eijk, A.M.J., Kunz, G.J., Dekker, H., Neele, F.P. en Schwering, P.B.W. (1995b). Atmospheric effects on detection of point targets at long ranges: literature study. TNO rapport, in voorbereiding.

[11] Edlén, B. (1966). The refractive index of air. Meteorology, 2, 71-80.

[12] Fenn, R.W., Clough, S.A., Gallery, W.O., Good, R.E., Kneizys, F.X., Mill, J.D., Rothman, L.S., Shettle, E.P. en Volz, F.E. (1985). Optical and infrared properties of the atmosphere, in: Handbook of Geophysics and Space Environment (A.S. Jursa, ed.), pp 18.1-18.80, Air Force Geophysics Laboratory, National Technical Information Service, Springfield, USA.

[13] Gathman, S.G. (1983). Optical properties of the marine aerosol as predicted by the Navy aerosol model. Optical Engineering, 22, 57-62.

[14] Gathman, S.G. (1989). A preliminary description of NOVAM, the Navy Oceanic Vertical Aerosol Model. NRL report 9200 (Washington D.C.).

[15] Gerber, H.E. (1985). Relative humidity parameterization of the Navy Aerosol Model (NAM). NRL report 8956 (Washington D.C.).

[16] Goroch, A., Burk, S. and Davidson, K.L. (1980). Stability effects on aerosol size and height distributions. Tellus, 32, 245-250.

[17] Hughes, H.G. (1987). Evaluation of the LOWTRAN 6 Navy maritime aerosol model using 8 to 12 μm sky radiances. Optical Engineering, 26, 1155-1160.

[18] Kneizys, F.X., Shettle, E.P., Gallery, W.O., Chetwynd Jr, J.H., Abreu, L.W., Selby, J.E.A., Clough, S.A. en Fenn, R.W. (1983). Atmospheric transmittance/radiance: computer code lowtran 6. Report AFGL-TR-83-0187.

[19] Kunz, G.J. (1993). On lidar signals induced by spatial variability of the atmospheric refractive index. TNO rapport FEL-92-A412.

[20] Kunz, G.J. (1996). A bulk model to predict turbulence in the marine surface layer. TNO rapport, FEL-96-A053.

[21] LeClerc, B. (1989). EO/IR Atmospheric transmittance model. Final report on contract 18SV.W8477-7-PF03. Defence Research Establishment Valcartier, Courcelette, Quebec, Canada.

[22] Liu, W.T., Katsaros, K.B. en Businger, J.A. (1979). Bulk parameterization of air-sea exchanges of heat and water vapor including the molecular constraints at the interface. Journal of the Atmospheric Sciences, 36, 1722-1735.

[23] Schwering, P.B.W. en van Eijk, A.M.J. (1994). IRST operational range prediction. TNO rapport FEL-93-A237 (confidentieel).

[24] Smith, S.D. (1980). Wind stress and heat flux over the ocean in gale force winds. Journal of Physical Oceanography, 10, 709-726.

[25] Smith, S.D. (1988). Coefficients for sea surface wind stress, heat flux, and wind profiles as a function of temperature. Journal of Geophysical Research, 93C, 15467-15472.

[26] Smith, S.D., Katsaros, K.B., Oost, W.A. en Mestayer, P.G. (1990). Two major experiments in the humidity exchange over the sea (HEXOS) program. Bulletin of the American Meteorological Society, 71, 161-172.

[27] Stull, B.R. (1988). An introduction to boundary layer meteorology. Kluwer, Dordrecht.

[28] Tanguy, M., Bonhommet, M., Autric, M. en Vigliano, P. (1991). Correlation between the aerosol profile measurements, the meteorological conditions and the IR transmission in a mediterranean marine atmosphere. Proceeding of SPIE, 1487, pp 17-1 to 17-12.

[29] Tennekes, H. (1973). A model for the dynamics of the inversion above a convective boundary layer. Journal of the Atmospheric Sciences, 30, 558-567.

[30] Tennekes, H. and Driedonks, A.G.M. (1980) Basic entrainment equations for the atmospheric boundary layer. Boundary Layer Meteorology, 20, 515-531.

[31] Van Eijk, A.M.J. en De Leeuw, G. (1992a). Modeling aerosol particle size distributions over the North Sea. Journal of Geophysical Research, 97C, 14417-14429.

[32] Van Eijk, A.M.J. en De Leeuw, G. (1992b). Modeling aerosol particle size distributions over the North Sea from HEXMAX data. TNO rapport FEL-92-A119.

[33] Van Eijk, A.M.J. en De Leeuw, G. (1992c). Modeling aerosol extinction in a coastal environment. Proceedings of SPIE, 1688, 28-36.

[34] Van Eijk, A.M.J., Bastin, F.H., Neele, F.P., De Leeuw, G. and Injuk, J. (1994). Characterization of Atmospheric properties during MAPTIP. The 2nd symposium of the SPP Panel of AGARD on 'Propagation assessment in coastal environments', 19-23 Sept 1994, Bremerhaven, Germany, CP-567.

8. Ondertekening

Drs. C.W. Lamberts
Groepsleider

Dr. A.M.J. van Eijk
Projectleider/Auteur

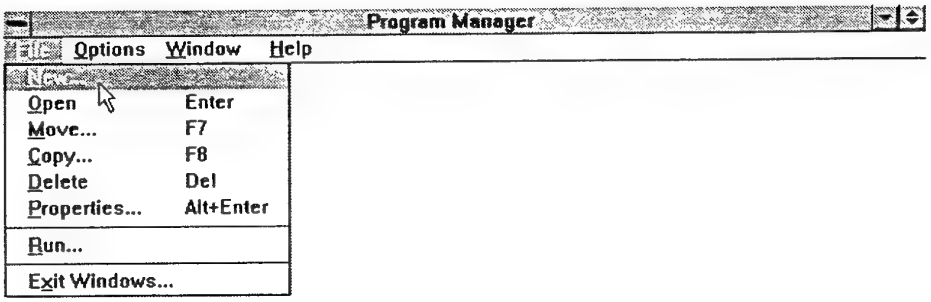
Bijlage A Installatie

Om RAID op het computersysteem te laten werken, moet het programma als volgt geïnstalleerd worden. Het is van groot belang om de aangegeven volgorde aan te houden zoals hier beschreven.

Begin allereerst in DOS met het volgende:

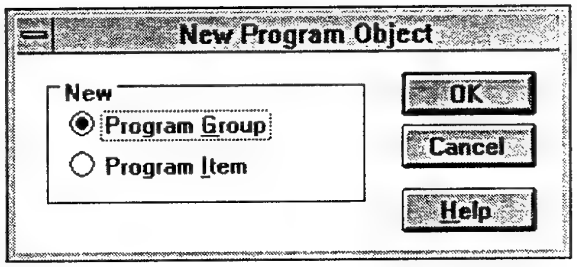
cd c:\
md RAID
cd RAID
copy a:*.exe c:

Hiermee is het DOS gedeelte afgerond, start nu Microsoft Windows. In Windows verschijnt een *desktop*, genaamd Program Manager. Kies voor de menu optie *File* en selecteer *New* (figuur A.1).



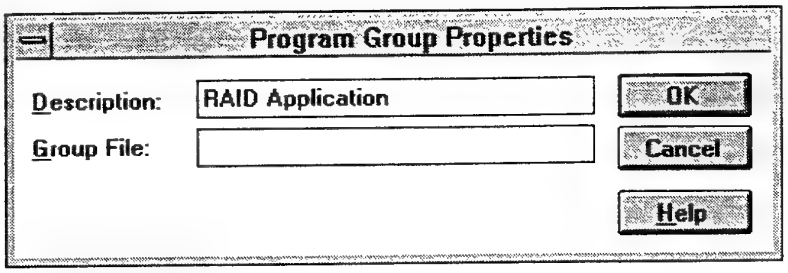
Figuur A.1: File New

Na het selecteren van *New* verschijnt er een dialogbox die vraagt of er een Program Group of een Program Item moet worden geplaatst binnen de *desktop*. Kies hier voor een *Program Group*. Figuur A.2 geeft dit aan.



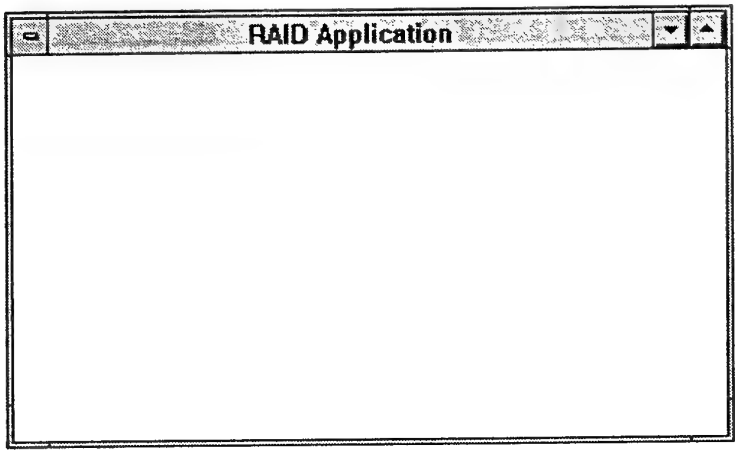
Figuur A.2: Program Group

Na het indrukken van de Ok button verschijnt een dialogbox waarin wordt gevraagd om de beschrijving van het programma. Voer hier in: RAID Application. Na invoeren ziet het dialogbox er uit als figuur A.3.



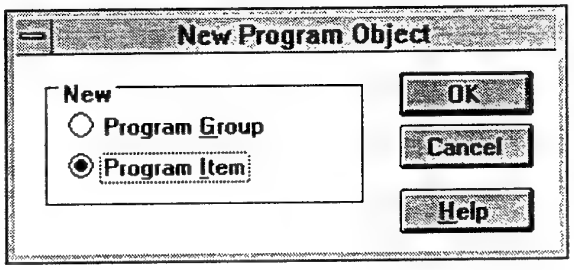
Figuur A.3: Beschrijving

Druk op OK en de dialogbox zal verdwijnen. Op de desktop verschijnt nu een lege group (weergegeven door een leeg window) met als naam : RAID Application. Het window ziet er uit zoals figuur A.4 aangeeft. Nu is de lege group toegevoegd.



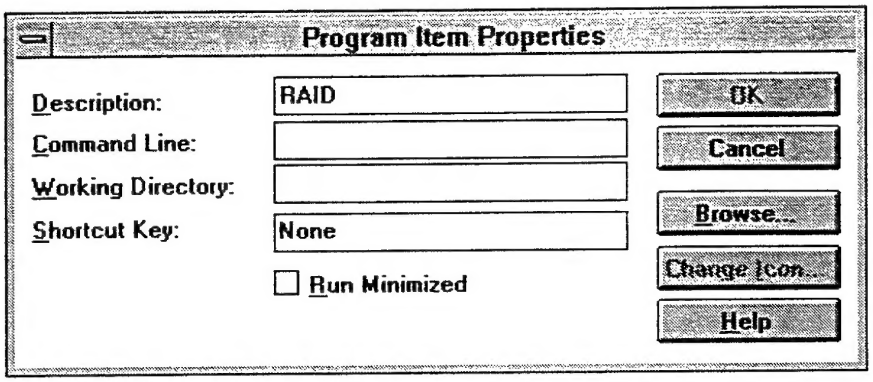
Figuur A.4: Lege group

Als volgende stap wordt opnieuw *New* in menu optie *File* gekozen, maar deze keer wordt *Program Item* geselecteerd, zoals aangegeven in figuur A.5.



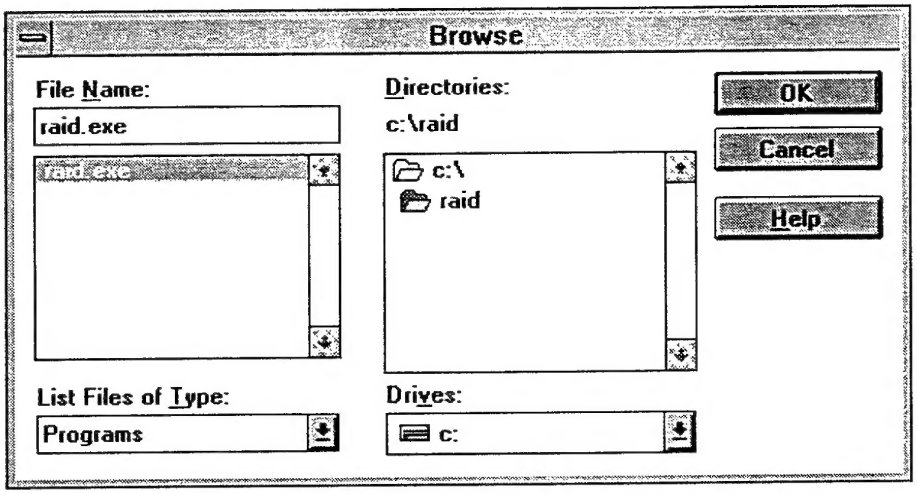
Figuur A.5: Program Item

Na het selecteren van *Program Item* en het drukken op de Ok button verschijnt een nieuwe dialogbox. Deze dialogbox moet ingevuld worden met de gegevens over het programma. Op de eerste regel (*Description*) staat de programma identificatie. Voer hier in: RAID (zie Figuur A.6).



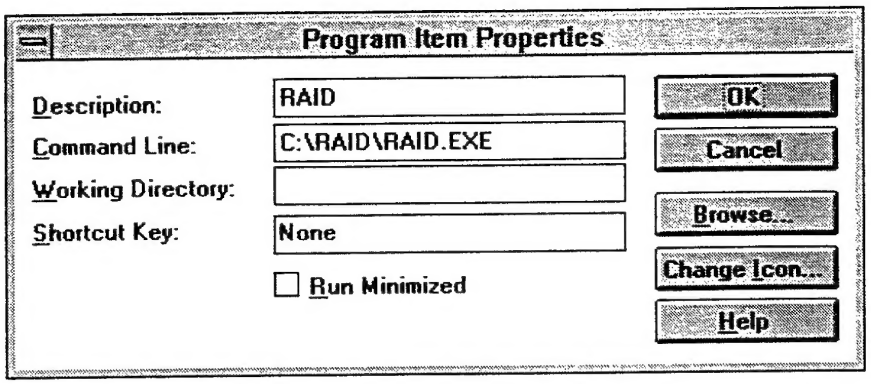
Figuur A.6: Programma gegevens

Vervolgens moet de koppeling worden gemaakt tussen de gegevens over het programma en het programma zelf. Dit wordt ingevoerd op de zogenaamde *Command Line* in de dialogbox van de Program Item Properties. Door *Browse* te kiezen verschijnt een nieuwe dialogbox zoals is aangeven in figuur A.7.



Figuur A.7: File dialogbox

Selecteer de directory c:\raid. Selecteer de file met als extensie *exe*. Zodra in de File Name-box hetzelfde staat als in de lijst druk dan op de Ok button. Hierna verdwijnt de dialogbox en is de *Command Line* binnen Program Item Properties gevuld. Het totaal ziet er uit als in figuur A.8.



Figuur A.8: Totale gegevens

Druk op Ok, het Program Item Properties gedeelte is afgerond.

Binnen het voorheen lege window staat nu een icon (figuur A.9). De program group zelf heeft ook een icon (figuur A.10) dat zichtbaar wordt als het window met minimize tot een icon wordt verkleind.

Figuur A.9: Raid icon

Figuur A.10: Group icon

REPORT DOCUMENTATION PAGE
(MOD-NL)

1. DEFENCE REPORT NO (MOD-NL)	2. RECIPIENT'S ACCESSION NO	3. PERFORMING ORGANIZATION REPORT NO
TD95-1648		FEL-95-A285
4. PROJECT/TASK/WORK UNIT NO	**5. CONTRACT NO**	**6. REPORT DATE**
22843	A92KM613	August 1996
7. NUMBER OF PAGES	**8. NUMBER OF REFERENCES**	**9. TYPE OF REPORT AND DATES COVERED**
83 (incl 1 appendix, excl RDP & distribution list)	34	Final

10. TITLE AND SUBTITLE

IR Reikwijdtevoorspeller - versie 1.0
(IR Range Predictor - version 1.0)

11. AUTHOR(S)

M.P. Sannes, J.C. Vuijst, Dr. A.M.J. van Eijk

12. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

TNO Physics and Electronics Laboratory, PO Box 96864, 2509 JG The Hague, The Netherlands
Oude Waalsdorperweg 63, The Hague, The Netherlands

13. SPONSORING AGENCY NAME(S) AND ADDRESS(ES)

DMKM/WCS/COSPON
Van der Burchlaan 31, 2597 PC The Hague, The Netherlands

14. SUPPLEMENTARY NOTES

The classification designation Ongerubriceerd is equivalent to Unclassified, Stg. Confidentieel is equivalent to Confidential and Stg. Geheim is equivalent to Secret.

15. ABSTRACT (MAXIMUM 200 WORDS (1044 BYTE))

The first version of an atmospheric module for an IR Range Predictor is described. A brief review of the various empirical and physical models that are implemented precedes an extensive user manual. In addition, the programming environment and some of the specific problems encountered during the program development are discussed. Recommendations for an improved module are given.

16. DESCRIPTORS	IDENTIFIERS
Infrared Radiation	Range Prediction
Marine Environments	Vertical Structure
Atmospheric Model	Refraction
Aerosol	

17a. SECURITY CLASSIFICATION (OF REPORT)	17b. SECURITY CLASSIFICATION (OF PAGE)	17c. SECURITY CLASSIFICATION (OF ABSTRACT)
Ongerubriceerd	Ongerubriceerd	Ongerubriceerd

18. DISTRIBUTION AVAILABILITY STATEMENT

Unlimited Distribution

17d. SECURITY CLASSIFICATION (OF TITLES)

Ongerubriceerd

Distributielijst

1. Bureau TNO Defensieonderzoek
2. Directeur Wetenschappelijk Onderzoek en Ontwikkeling*)
3. HWO-KL*)
4. HWO-KLu*)
5. HWO-KM*)
6. HWO-CO*)

7 t/m 9. KMA, Bibliotheek

10. KMA, t.a.v. Ir. J. Rogge
11. KIM, Bibliotheek
12. KIM, t.a.v. Prof. Absil
13. DMKL/AB, t.a.v. Dr. I. van Ittersum
14. DMKL/BEWAPENING, t.a.v. Ing. G. Zwiep
15. DOPKLu/METEO
16. DMKM/HEMDC
17. KNMI, t.a.v. Dr. A. Lammeren
18. HTS Haarlem, t.a.v. studierichtings secretariaat
19. HTS Haarlem, t.a.v. Ing. M.P. Sannes
20. HTS Haarlem, t.a.v. Ing. J.C. Vuijst
21. DMKM/WCS/COSPON, t.a.v. Drs. W. Pelt
22. Dienst der Hydrografie, t.a.v. LTZ1 C. de Looze
23. Directie TNO-FEL, t.a.v. Dr. J.W. Maas
24. Directie TNO-FEL, t.a.v. Ir. J.A. Vogel, daarna reserve
25. Archief TNO-FEL, in bruikleen aan M&P*)
26. Archief TNO-FEL, in bruikleen aan Ir. J. Bennema
27. Archief TNO-FEL, in bruikleen aan Ir. C. Eberwijn
28. Archief TNO-FEL, in bruikleen aan Drs. C.W. Lamberts
29. Archief TNO-FEL, in bruikleen aan Dr. ir. H.M.A. Schleijpen
30. Archief TNO-FEL, in bruikleen aan Dr. G. de Leeuw
31. Archief TNO-FEL, in bruikleen aan Dr. P.B.W. Schwering
32. Archief TNO-FEL, in bruikleen aan Dr. A.M.J. van Eijk
33. Archief TNO-FEL, in bruikleen aan Ir. G.J. Kunz
34. Archief TNO-FEL, in bruikleen aan Drs. B.J. van Dam
35. Documentatie TNO-FEL
36. Reserve

TNO-PML, Bibliotheek**)
TNO-TM, Bibliotheek**)
TNO-FEL, Bibliotheek**)

Indien binnen de krijgsmacht extra exemplaren van dit rapport worden gewenst door personen of instanties die niet op de verzendlijst voorkomen, dan dienen deze aangevraagd te worden bij het betreffende Hoofd Wetenschappelijk Onderzoek of, indien het een K-opdracht betreft, bij de Directeur Wetenschappelijk Onderzoek en Ontwikkeling.

*) Beperkt rapport (titelblad, managementuittreksel, RDP en distributielijst).
**) RDP.